EFFICIENT RANGING-SENSOR NAVIGATION METHODS FOR INDOOR AIRCRAFT

A Thesis
Presented to
The Academic Faculty

by

David Michael Sobers, Jr.
Major, USAF

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Aerospace Engineering in the
Daniel Guggenheim School of Aerospace Engineering

Georgia Institute of Technology
August, 2010

EFFICIENT RANGING-SENSOR NAVIGATION METHODS FOR INDOOR AIRCRAFT

Approved by:

Eric N. Johnson, Advisor
Lockheed Martin Associate Professor of
Avionics Integration
School of Aerospace Engineering
Georgia Institute of Technology

Mark Costello
Sikorsky Associate Professor
School of Aerospace Engineering
Georgia Institute of Technology

Eric Feron
Dutton/Ducoffe Professor of Aerospace
Software Engineering
School of Aerospace Engineering
Georgia Institute of Technology

Frank Dellaert
Associate Professor
School of Interactive Computing
Georgia Institute of Technology

Date Approved: June 25, 2010

Carlee Bishop
Senior Research Engineer
Georgia Tech Research Institute

DEDICATION

This work is dedicated to my family, who supported me through all the long hours, missed dinners, and nights I worked late. To my wife, Christy, who was with me every step of the way – this work could not have been completed without your love and support. To Nathan, Allison, and Lily – you have your Dad back now. Thanks for understanding when I had to say, "sorry, I can't play now, I have to finish writing." I also dedicate this work to my Lord and Savior Jesus Christ, who gives me wisdom, courage, and strength when I need it most.

ACKNOWLEDGEMENTS

Each of my committee members has contributed in an instrumental way to this research by providing advice on both practical and theoretical aspects of the research, through helpful course lectures, and offering professional advice in general. Thanks especially to Dr. Johnson, who was also my academic advisor. To each of the committee members I also offer my gratitude: Dr. Mark Costello (AE), Dr. Eric Feron (AE), Dr. Carlee Bishop (GTRI), and Dr. Frank Dellaert (IC/CoC). This research would also have been impossible without the help of the 2009-2010 Georgia Tech Aerial Robotics (GTAR) teams. The GTAR members are all volunteers and spent numerous hours during their "free" time working on the project in preparation for the International Aerial Robotics Competition. The team members include (in alphabetical order): Girish Chowdhary, Claus Christmann, Hiroyuki Hashimoto, Scott Kimbrell, John Ottander, Ep Pravitra, Richard Roberts, Dr. Erwan Salaün, Syed Shah, Allen Wu, and Shusaku Yamaura. Thanks also to Jeong Hur for help and advice during construction of the vehicles.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

A, B	Linear regression parameters used in Polar Recursive Least Squares estimation
D	Perpendicular distance from the vehicle to the wall
$\mathbf{F}$	Jacobian matrix that is the partial derivative of the nonlinear system dynamics with respect to the states
$\mathbf{H}$	Jacobian matrix that is the partial derivative of nonlinear measurements with respect to the states
$\mathbf{I}$	Information matrix
$\mathbf{K}$	Kalman gain matrix
K_p, K_i, K_d	Proportional, integral, and derivative gains used for feedback control
K_v	Proportional velocity gain for guidance system commanded velocity
L	Infrared range sensor lateral separation
$\mathbf{P}$	Covariance matrix of the state estimate
$\mathbf{Q}$	Covariance matrix of the plant noise
$\mathbf{R}$	Covariance matrix of the measurement noise
$\mathbf{\Sigma}_r^2$	Diagonal matrix of laser scan range variances
$\mathbf{v}$	Measurement noise, assumed zero mean Gaussian distribution
$\mathbf{w}$	Plant noise, assumed zero mean Gaussian distribution
$x_{_L}$, $x_{_R}$	Left and right infrared range sensor readings
$\mathbf{x}$, $\dot{\mathbf{x}}$	State vector, and its time rate of change
$\hat{\mathbf{x}}$, $\dot{\hat{\mathbf{x}}}$	Estimate of the state vector, and it time rate of change
$\mathbf{z}_k$	Measurement vector
γ_i	Incremental commanded velocity (summed over scan points)
σ_r	Range measurement standard deviation
ψ	Vehicle heading
(ρ,ϕ)	Line parameters in normal form
(r,θ)	Polar coordinate pair
(x,y)	Cartesian coordinate pair
(x,y,θ)	Vehicle pose, corresponding to the 2-D position and heading
$(\Delta x,\Delta y,\Delta\theta)$	Vehicle pose error, calculated by the SLAM navigation algorithms

LIST OF ABBREVIATIONS

DCM	Direction Cosine Matrix
DOP	Dilution of Precision
EKF	Extended Kalman Filter
GCS	Ground Control Station
GNC	Guidance, Navigation, and Control
GPS	Global Positioning System
IARC	International Aerial Robotics Competition
ICP	Iterative Closest Point
IEPF	Iterative End Point Fit
IMU	Inertial Measurement Unit
IR	Infrared
LT	Line Tracking
NED	North - East - Down
PF	Particle Filter
PID	Proportional/Integral/Derivative
PRLS	Polar Recursive Least Squares
PW	Pulse Width
SLAM	Simultaneous Localization and Mapping
SM	Split and Merge
UAV	Unmanned Aerial Vehicle

SUMMARY

Unmanned Aerial Vehicles (UAVs) are often used for reconnaissance, search and rescue, damage assessment, exploration, and other tasks that are dangerous or prohibitively difficult for humans to perform. Often, these tasks include traversing indoor environments where radio links are unreliable, hindering the use of remote pilot links or ground-based control, and effectively eliminating Global Positioning System (GPS) signals as a potential localization method. As a result, any vehicle capable of indoor flight must be able to stabilize itself and perform all guidance, navigation, and control (GNC) tasks without dependence on a radio link, which may be available only intermittently.

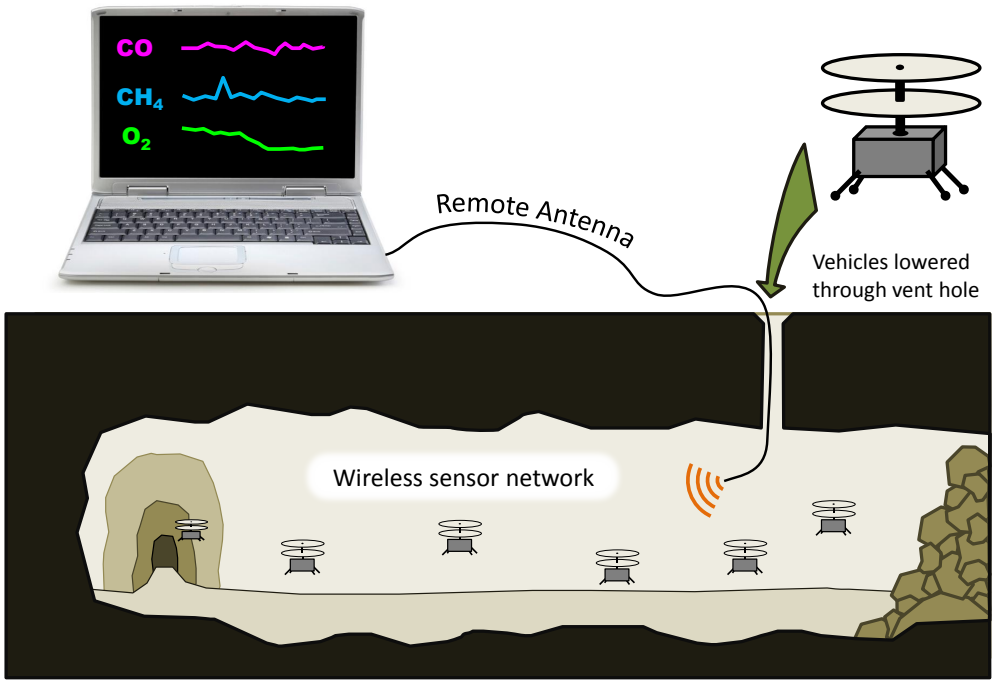
Stability and control of rotorcraft UAVs is usually achieved by either a passive stability system, such as a Bell stabilizer bar, or by actively measuring body accelerations and angular rates with an onboard Inertial Measurement Unit (IMU) and using that data for feedback control. However, neither active nor passive attitude stabilization methods provide position control by themselves. Therefore, GNC methods must either be tolerant to position drift or have some means of estimating and controlling position, which requires an external reference in order to measure and correct errors in the position estimate. GPS signals are often the most convenient method for providing this external position reference. As a result, most UAVs utilize GPS for localization and to bound error on position drift.

Unfortunately, the availability of GPS signals in unknown environments is not assured, especially during indoor operation. As a result, other sensors must be used to provide position information relative to the environment. This research covers a description of different ranging sensors and methods for incorporating them into the overall guidance, navigation, and control system. Various sensors are analyzed to determine their performance characteristics and suitability for indoor navigation, including sonar, infrared (IR) range sensors, and a scanning laser rangefinder. Each type of range sensor tested has its own unique characteristics and contributes in a slightly different way to effectively eliminate the

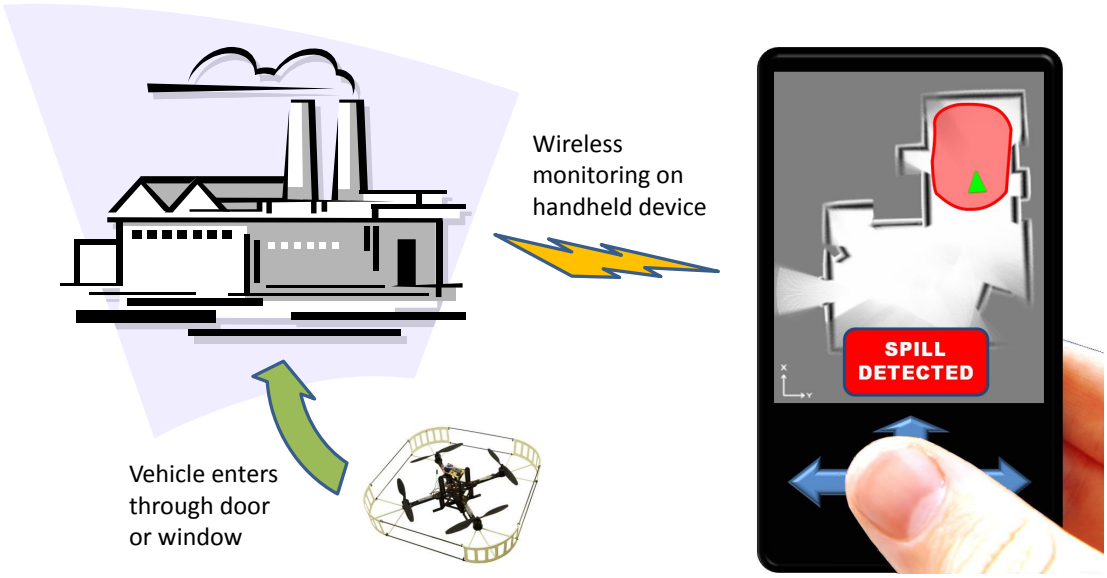
dependence on GPS. A discussion of vehicle selection criteria is provided, with an analysis of the pros and cons associated with passively versus actively stabilized vehicles. Two levels of navigation capability are presented, each providing different levels of mission performance.

First, the use of low-cost range sensors on an inexpensive passively stabilized coaxial helicopter for drift-tolerant indoor navigation is demonstrated through simulation and flight test. The system developed and flown shows promise for future miniaturization, enabling vehicles to be easily transported in large quantities to remote areas. One potential use for such a vehicle would be the distribution of small lightweight sensors deep into a mine or cave through a small opening. Figure 1(a) shows an example of such a deployment. Other sensors, including video cameras and microphones, could also be used in a variety of similar scenarios where access is limited.

For situations where some level of map building may be required, as shown in Figure 1(b), additional capabilities are needed. Hence, this thesis also describes a system with a scanning laser rangefinder mounted onboard a quadrotor helicopter with an IMU to enable active stabilization. Position control is demonstrated in simulation via navigation algorithms that utilize Simultaneous Localization and Mapping (SLAM) techniques. Two different algorithms are evaluated for suitability for use with an IMU-stabilized flying vehicle. Simulation and experimental results of the navigation system are provided, including two novel approaches to extracting useful information about the environment from laser scan data. Thus, the research presented in this thesis demonstrates new capabilities for autonomous UAV operation in adverse unknown environments, specifically those in which GPS signals are unavailable.



(a) Low-cost miniature vehicles collect air samples in a collapsed mine



(b) Small vehicle with basic mapping locates a hazardous spill in a chemical plant

Figure 1: Concepts of Operation. Different levels of autonomous navigation capability may be needed depending on the mission requirements.

CHAPTER I

INTRODUCTION

This research describes several methods by which Unmanned Aerial Vehicles (UAVs) can autonomously navigate unknown indoor environments. Techniques are discussed for guidance, navigation, and control (GNC) of rotorcraft vehicles. Both passively and actively stabilized rotorcraft vehicle platforms are considered, and different GNC methods are explored for different vehicle and sensor combinations. Sensors include sonar, infrared range sensors, and a scanning laser rangefinder, as well as an integrated inertial measurement unit (IMU) for acceleration and angular rate measurement. For passively stable rotorcraft, a basic navigation algorithm using only range information was developed, simulated, and flown experimentally. Actively stabilized vehicles require a continuous estimate of vehicle position and velocity through the integration of IMU data. The state estimate drifts over time, so range sensors are used to bound this drift enabling position control relative to the stationary environment. In addition, the accumulation of range data over time can be used to develop and maintain a map of the environment. The vehicle can then localize itself using its map of the environment. This process is commonly known as the Simultaneous Localization and Mapping (SLAM) problem. Two different SLAM algorithms are evaluated, with simulation results and experimental test results presented.

1.1 Motivation

Autonomous mobile robots that can effectively navigate unknown environments could be utilized for a wide range of applications, including search and rescue, disaster assessment, reconnaissance, or other tasks that would be risky or impossible for a human to perform. However, there are several technical challenges that hinder reliable operation of UAVs in these environments. Situational awareness must be maintained to prevent collision with obstacles and to ensure stable flight. This is particularly important for autonomous vehicles, where an operator is not present to provide input to the vehicle. For simple autonomous

exploration missions, a low-cost passively stabilized vehicle with basic guidance laws may be useful. In cases where more complex behavior is required, a UAV may rely on some method of mapping the unknown environment and determining its location within the environment in order to accomplish its mission. Most UAVs utilize Global Positioning System (GPS) signals for localization. In addition to aiding navigation, the GPS position measurement is used to bound drift in position estimates caused by the integration of acceleration and angular rate sensors. Since the availability of GPS signals indoors is not assured, other sensors must be used to provide position information so that a vehicle can make corrections to its state estimate. An aerial vehicle that can perform simultaneous localization and mapping using range sensors would thus be capable of navigating within unknown indoor environments where GPS signals are not available.

1.2 Aerial Vehicle Options

Indoor flight usually requires navigating small spaces with obstacles in close proximity. As a result, only vehicles with sustained hover capability were evaluated during this research. Fixed-wing aircraft require considerable space to maneuver, and one small enough to maneuver in a typical indoor environment would likely be unable to carry a useful payload or the necessary sensors and computers required to perform SLAM given the current state of the art. Hence, in this research simulation and flight testing was accomplished using rotorcraft, although the techniques developed during this research could in theory be applied to fixed-wing aircraft flying in outdoor GPS-denied areas as well. Lighter-than-air vehicles were not considered, due to their low maneuverability and limited payload capacity.

Although the capability to hover is an advantage for indoor flight, a hovering vehicle is typically unstable and requires either passive or active stabilization. Passive and active stabilization methods each have advantages and disadvantages, and each system has unique challenges for indoor navigation. A passively stabilized system, such as a coaxial helicopter with a Bell stabilizer bar, can be controlled using only range sensors for position control [59]. Thus, the guidance, navigation, and control (GNC) algorithms are simplified compared to those for an actively stabilized system. Unfortunately, the same mechanism

that enables passively stable hovering flight also limits maneuverability. Actively stabilized vehicles with hover capability include traditional single-rotor helicopters, quadrotors and other multi-rotor vehicles, and ducted fan vehicles. Although these vehicles are generally more maneuverable, active stabilization requires the measurement of body acceleration and angular rate, usually through the use of an IMU. As discussed in the following chapters, this research includes the design, construction, and testing of a passively stable coaxial helicopter capable of simple indoor navigation, and the use of an actively stabilized quadrotor vehicle to test more advanced navigation techniques.

1.3 Sensor Options

Several different sensors were evaluated for use in indoor navigation during this research. Initial emphasis was on lightweight, low cost sensors that could be used for ranging and obstacle avoidance. Sonar is a good choice for wide area coverage, and when longer range measurements are required. Infrared range sensors have a narrower beam width and shorter range, but unlike sonar they do not suffer from multi-path errors and they do not tend to interfere with each other during simultaneous operation. A scanning laser rangefinder provides a wide field of view 2-dimensional range map, which can be used to determine vehicle 2-D position and heading relative to a static environment. For aerial vehicles that require active stabilization, an IMU is necessary to measure vehicle acceleration and angular rate. In addition to the sensors mentioned here, other sensors not used in this research that may also be used to estimate vehicle motion include machine vision, optical flow sensors, 3-D scanners, and time-of-flight cameras, to name a few. Further information on specific sensors and how they were utilized in the research is provided in the chapters below.

1.4 Guidance, Navigation, and Control

When using a passively stabilized vehicle to perform indoor missions, the task of guidance, navigation, and control is greatly simplified. Algorithms can utilize small, lightweight ranging sensors to measure distance to obstacles in the environment. GNC algorithms can then be tailored to perform basic flight behaviors using only relative information without generating an inertial position estimate [59, 22, 52].

For more complex missions, an actively stabilized vehicle may be useful. An IMU can be used onboard the vehicle to measure body accelerations and angular rates, which are then integrated to estimate velocity and position. The IMU data, along with other measurements, are typically used with a Kalman filter to maintain an estimate of the vehicle state. A position measurement is also required to bound the drift that is inherent with the estimation process. In lieu of GPS signals, a scanning laser rangefinder can be used to estimate vehicle motion and relative position [1].

1.5 Review of Relevant Literature

Navigation of autonomous vehicles in unknown environments has been a topic of study in the field of robotics for over two decades, especially as applied to ground vehicles. The work published by Self, Smith, and Cheeseman [58] and the work by Leonard and Durrant-Whyte [37] is generally considered the first attempt to solve the problem of estimating vehicle position from observation of landmarks in the environment. Solving the problem of "Where am I?" is the primary topic of Leonard's book [36], and it requires in its most basic form a moving vehicle, a sensor that can measure distance (angular or linear) to some feature in the environment, and some way to correlate current measurements with past measurements. The problem, now known as Simultaneous Localization and Mapping (SLAM), was first proposed and solved using ground-based vehicles with sonar range sensors [36]. Since its inception, solving the SLAM problem has evolved and matured to include a wide variety of vehicle platforms, sensor packages, and many different algorithms for localization and mapping [51, 6, 7, 23, 24, 64].

Early SLAM algorithms utilized an Extended Kalman Filter (EKF) to keep track of the position of landmarks observed in the environment. A general overview of current EKF-SLAM theory, formulated without respect to the specific platform or sensors, can be found in [17, 3]. More recent work using a nonlinear observer to estimate landmark and vehicle states can be found in [66].

As scanning laser sensors became more widely available, methods were developed to

use the entire scan data instead of identifying specific landmarks. Development of scan-matching algorithms is well documented in [38, 40, 15, 46]. Later efforts successfully married the techniques of EKF-SLAM and scan-matching for even more effective algorithms [45, 8]. More recent developments include solutions which utilize rich 3-dimensional data sets [35, 72, 53, 54, 39] and attempt to map dynamic environments [25, 68].

The task of correlating noisy sensor readings with uncertain vehicle motion to estimate vehicle position has also become less daunting over the years due to advancements in sensing technology. In addition to sonar mentioned previously, radar [16, 32], scanning laser range finders [20], and vision sensors [42, 29, 57, 34] have been successfully used for mapping and localization on ground based vehicles. A thorough survey of vision-based SLAM techniques can be found in [11].

As the use of laser scanners for localization has increased in popularity, techniques for analyzing the feature-rich data they generate have been widely studied. In fact, some of the faster and more popular techniques for extracting line segments from laser scan data began in the 1970's [49]. Feature extraction from laser scan data, including line segments, corners, and curves, is performed by many different methods [2, 5, 43, 67, 50, 63, 65]. An excellent overview of different algorithms and their comparative computational complexity and performance was performed by Nguyen et al in [44].

Each SLAM method has its own advantages and disadvantages, and any robust system will likely utilize a combination of different sensors and algorithms. For air vehicles, however, methods are somewhat restricted due to vehicle constraints. Air vehicles have limited payload capacity, so sensors and onboard computing power are limited. Vehicle motion is also more complex, requiring more complex state estimation algorithms. The dynamics of an airborne vehicle span six degrees of freedom, and are often faster and more difficult to model than common ground vehicles. As a result of these challenges, airborne vehicles have only recently been explored as platforms for SLAM experimentation.

Early airborne SLAM work utilized fixed-wing vehicles to map terrain landmarks of known size and shape using vision and radar sensors [32]. In this work, the SLAM navigation solution was evaluated by comparing it with a combined IMU/GPS solution. Later, IMU

data was integrated with the SLAM algorithm in a real-time implementation as described in [33, 9]. When compared to GPS-corrected position data, the vehicle was able to determine its position with 1-σ values ranging from 5-20 meters. While these results demonstrated that SLAM is a viable method for airborne navigation, the vehicle used and position accuracy achieved fall short of what would be required for successful indoor navigation.

More recently, vision-based localization and mapping has been applied to helicopter platforms as described in [70, 69, 10]. Due to their unstable flight characteristics, helicopters require a much faster attitude control system to maintain flight. Hence, they are usually stabilized by use of an IMU for rate feedback control, with some type of localization required to prevent position drift. For indoor flight of unstable vehicles, localization must be highly accurate with fast update rates to avoid collision with the environment. A minimal system for autonomous indoor flight was demonstrated in [52] using a quadrotor with an IMU for active attitude stabilization and infrared range sensors for obstacle detection. This system did not demonstrate strict position control, but was primarily concerned with obstacle avoidance. The preliminary results of the research presented in this thesis, as documented in [59], achieved position control and limited indoor navigation by using a passively stabilized coaxial helicopter with only range sensors for relative localization. The system was fully autonomous with a control station only required for observation. However, no mapping was attempted, primarily due to the limited sensor suite and computational power onboard.

Perhaps the most advanced demonstration of indoor navigation to date is the work described in [1]. This work uses the techniques developed in [48] to successfully integrate a laser scanner and stereo vision system with an IMU onboard a quadrotor helicopter. The system is capable of localization and mapping within unknown indoor environments, including autonomous exploration and path planning, utilizing a suite of ground station computers to process laser and vision data and to perform SLAM algorithms. An Extended Kalman Filter is used to estimate the vehicle state at the ground station, with vehicle commands provided to the onboard computer over a Wi-Fi radio link. While this system demonstrates complex mapping and localization during autonomous flight, the dependence

on ground computers and a reliable data link could prevent the vehicle from exploring deep into caves or structures. In such situations, it is desirable for a vehicle to have no dependence on external resources.

The research presented in this thesis approaches the design of the navigation system from the perspective that all necessary sensors and computational power should be carried onboard the vehicle. It is this design philosophy which has led to contributions on two different fronts. First, a very minimal navigation system consisting of only five range sensors was designed, implemented, and flown onboard a passively stable coaxial helicopter. Second, a more capable navigation system utilizing a scanning laser rangefinder for localization and mapping was designed and tested in simulation and experimentally. This laser-aided navigation system also utilizes an IMU and a sonar, with an EKF navigation filter providing state estimates, all designed to run onboard a small embedded computer capable of being flown on a small quadrotor vehicle.

1.6 Contributions to the Field

The chapters that follow describe in detail the contributions made by this research in the areas of guidance and navigation systems for autonomous indoor flying vehicles utilizing range sensors for position estimation. As the primary focus is on addressing problems related to navigation in GPS-denied environments, however, the guidance systems described herein are primarily designed around demonstrating novel concepts for indoor navigation. Specific contributions in the area of indoor navigation are summarized as follows:

- Demonstration of autonomous indoor navigation utilizing low-cost range sensors on a passively stabilized helicopter, with results suggesting the miniaturization of this method to vehicles weighing less than 50g is feasible
- Development of an algorithm for extracting line segments from laser scan data using a polar form recursive least squares approach that is an order of magnitude faster than previous similar methods
- Development of an Iterative Closest Point scan matching algorithm that utilizes key

frame scans to reduce the rate of position estimate drift compared to sequential scan-matching techniques

- Development of a method for determining pose estimate uncertainty based on the shape of the features observed in the environment
- Analysis of two fast SLAM algorithms for pose estimation using a scanning laser rangefinder, and testing of both methods in simulation and experimentally to determine suitability for indoor flight navigation
- Demonstration that these "minimum" capabilities performed on board a vehicle enable a completely self-contained autonomous UAV to successfully navigate indoors

1.7 Organization of the Thesis

The research presented in this thesis consists of two different approaches to solving the problem of indoor navigation for autonomous flying vehicles. The organization of the thesis is thus built around the presentation of these two solutions. The introduction in this chapter provides a description of the problems associated with navigating in indoor and other GPS-denied environments, and covers the motivation behind the development of new approaches to solving these problems. An overview of the types of vehicles and sensors that are suitable for indoor flight is included, as well as a specific challenges faced when attempting guidance, navigation, and control in indoor environments. A review of relevant literature is provided, covering several decades of research in GNC of unmanned aerial vehicles as well as navigation techniques commonly used for autonomous ground vehicles. Finally, contributions in the the field of indoor aerial navigation are specifically listed, with more details provided in the following chapters.

Before discussing the research in detail, background information on the problem is provided in Chapter 2. A discussion of guidance, navigation, and control techniques is provided, with an emphasis on the special considerations required for indoor flight. Next, a formulation of the Extended Kalman Filter with continuous propagation and discrete updates is

provided. Finally, Chapter 2 covers an overview of the SLAM problem and how aerial navigation differs from traditional ground-based algorithms. Two classic approaches to solving the SLAM problem are summarized, with a discussion of which approach is best suited for laser-aided indoor aerial navigation. A description of the information that can be extracted from laser scan measurements is provided at the conclusion of the discussion on SLAM.

Following the background information, Chapter 3 covers a simple, low-cost approach to indoor navigation. The choice of a coaxial helicopter fitted with infrared and sonar range sensors is discussed in detail, followed by the specific guidance, navigation, and control algorithms implemented. The system described in Chapter 3 is designed around the use of small, lightweight, inexpensive range sensors for navigation relative to the environment. This minimal system is capable of providing estimates of lateral position, longitudinal position, heading, and altitude all relative to the local environment. Rate of change of these quantities is also estimated, providing all the necessary information to the guidance and control algorithms to perform autonomous indoor flight. A complete description of the design, implementation, and flight testing of the navigation system is provided in Chapter 3. The system has the potential for easy miniaturization to vehicles as small as 50g, providing a significant contribution toward the development of extremely small fully autonomous aerial vehicles.

A more capable system, which is also more complicated and more expensive, is described in Chapter 4. This navigation system is designed to provide not only vehicle localization, but active stabilization of an unstable rotorcraft vehicle through the fusion of measurements from an IMU, a sonar altimeter, and a scanning laser rangefinder in an EKF navigation architecture. Two specific SLAM algorithms are discussed in detail, along with specific techniques for extracting useful information from laser scan data. Guidance and navigation algorithms that utilize these sensors are described in detail, followed by a discussion of the simulation environment. Two navigation algorithms are compared in several identical scenarios to determine the advantages and disadvantages of each. Finally, a description of experimental tests using a laser scanner, IMU, and the EKF navigation system is provided, with results using the two different SLAM algorithms.

After the discussion of the different approaches to indoor navigation provided in Chapters 3 and 4, a summary of the conclusions is given in Chapter 5 with a description of the contributions made as a result of this research. Following the conclusions, references are provided, as well as an appendix that contains a theorized but untested method for using sonar range sensors to perform mapping on a stable flight vehicle as an intermediate level of complexity between the systems described in Chapters 3 and 4.

CHAPTER II

BACKGROUND

2.1 UAV Guidance, Navigation, and Control

The ability for any vehicle to move about in a way that progresses toward some goal depends heavily on three things: a measure of the current state of progress toward the goal, a plan for continuing or improving that progress, and a method for effecting required changes to follow the aforementioned plan. To these ends, the terms Navigation, Guidance, and Control are used here, respectively. This chapter briefly discusses the role of each in the ability for an air vehicle to successfully maneuver in an unknown indoor environment without any external inputs. Often, the term *indoor navigation* will be used throughout as a general description of the entire Guidance, Navigation, and Control system and its ability to effect desired vehicle motion. The following subsections will describe how specialization to indoor flight affects the design of the GNC system for both stable vehicles with simple avionics packages, and unstable vehicles with more advanced avionics.

2.1.1 Guidance Philosophy

During any mission, an autonomous vehicle must make decisions about what behavior is required for success. The *guidance system* encompasses that decision making process, which may be as simple as a hierarchy of rules to follow. In order for the guidance system to work effectively, it must have a measure of the vehicle's current status with respect to mission goals (provided by the navigation system) and a means to change the vehicle's motion if required (executed through the control system). For indoor navigation, typical goals include obstacle avoidance, exploration, target identification, and other surveillance or reconnaissance activities. Depending on the complexity of the mission, these goals may be met by simple random flight that avoids obstacles, or an advanced system of mapping and path planning to maximize search effectiveness may be required. The complexity of the guidance system, as determined by mission requirements, heavily influences the complexity

of the navigation and control systems. This research focuses on indoor flight at two different levels of complexity, and provides significant contributions through the complete design, simulation, and testing of the guidance systems and the navigation and control systems required for their implementation.

An important aspect of the guidance system is to determine which direction and how fast a vehicle should fly in order to meet its goals. Traditionally, flying vehicles designed for outdoor use rely on inertial navigation (via GPS-aided IMU) and as a result position and velocity goals are typically determined with respect to an inertial reference frame. Complex path planning and trajectory generation algorithms have been developed and implemented with great success using this framework. For indoor flight, however, directions like *north* or *east* may have little or no meaning (although *down* is still fairly universal). Without an inertial position reference like GPS, the guidance system requires a very different approach. Position and velocity goals must be established with respect to a local reference frame which has an unknown relationship to the inertial frame. The chapters that follow discuss in detail the guidance systems developed for indoor flight during this research.

2.1.2 Navigation Philosophy

In order for a vehicle's guidance system to determine what position and velocity goals are required, a reasonable estimate of the vehicle's state with respect to those goals is needed. The *navigation system* provides this state estimate, based on sensor inputs and some algorithm to extract useful information from them. Naturally, the complexity of the navigation system and the state estimate produced is influenced by the mission requirements and the plan to achieve them, as well as the type of information required by the control system to determine correct actuator inputs. A major contribution of this research is the development and testing of a low-cost, lightweight navigation system for autonomous flight of a passively stable flying helicopter, as well as the design of a minimal system for a vehicle that requires active stabilization. Here, the focus is on determining the lower limit for successful indoor navigation given two different air vehicle designs, rather than throwing the massive resources available in the robotics community at the problem. It is

an approach that seeks to minimize weight and computational resources required by the navigation system, leaving more capacity available for mission related sensors. Unlike efforts to date [1], these navigation algorithms and the required sensors for each have been designed for implementation on completely self-contained autonomous vehicles that do not require any external navigation support.

Passively stabilized aircraft are capable of limited indoor flight behaviors using simple systems, however navigation systems for vehicles that require active flight stabilization are necessarily more complex. In addition to providing position and velocity estimates, the vehicle attitude and angular rates must also be estimated and controlled. Hence, a full inertial measurement unit is required to sense body accelerations and angular rates and the vehicle state must be estimated using those measurements. An effective method for GNC of inherently unstable flying vehicles is to use an Extended Kalman Filter to produce a state estimate, with associated uncertainty, from measurements produced by an IMU [14]. This estimate will drift over time if no external position reference is used to make corrections. Typically, GPS signals are used to provide the position measurements necessary to bound the drift in the navigation solution.

For indoor flight, another localization method is required in lieu of GPS, since the satellite signals do no penetrate well into building structures or underground. A localization method commonly used on ground vehicles is the scanning laser rangefinder. Using a laser scanner to provide 2-D position and heading is a well-studied problem, with many algorithms freely available within the computer science and robotics community. However, these algorithms have evolved through many years of experimentation on ground-based vehicles. Ground vehicles have much slower and more stable dynamics than an air vehicle, they can carry greater computational power, and they often perform longer missions covering greater area than an indoor flying vehicle would. As a result, many of the algorithms developed for ground vehicles to analyze laser scan data for localization and mapping are not ideal for air vehicles using current embedded computer systems due to their computational requirements. Any navigation algorithm must be able to process scan data in real time with resources shared between flight stabilization, GNC, and other mission related

processes. Past efforts have solved this problem by offloading the guidance and navigation tasks, including analysis of laser scan data, to a network of computers on the ground [1].

In contrast, this research provides contributions in the area of finding navigation solutions that are compatible with completely self-contained autonomous indoor flight using currently available hardware. To that end, several methods for analyzing laser scan data were implemented and tested for suitability with regard to the computational power available on typical lightweight embedded computer systems. Research efforts focused on finding, evaluating, and optimizing scan matching algorithms specifically for indoor flight. The scan matching algorithms were used to estimate the vehicle's 2-D position and heading, also known as the vehicle pose (x, y, θ).

Many techniques have been developed for estimating states of systems with nonlinear dynamics [21, 66].

Pose estimates were incorporated with an existing UAV navigation filter that is designed to take external measurements as inputs to the filter.

The existing navigation filter is based on an Extended Kalman Filter with continuous state and covariance propagation and discrete measurement updates. The EKF is described in more detail below, along with an overview of Simultaneous Localization and Mapping techniques. Details on the laser-aided inertial navigation system developed during this research, including design, implementation and experimental results, are provided in Chapter 4.

2.1.3 Control Philosophy

Given a reasonable estimate of the vehicle's current state, the guidance system determines what actions should be taken in order to achieve the mission and set intermediate position and/or velocity goals as required. It is the job of the *control system* to determine what instructions to give the actuators in order to minimize the error between the commanded state and the measured state. The complexity of the control system depends not only on the quantities being controlled, but also on the dynamics of the system itself. Unlike ground vehicles, unstable air vehicles are susceptible to oscillation and divergent flight when the

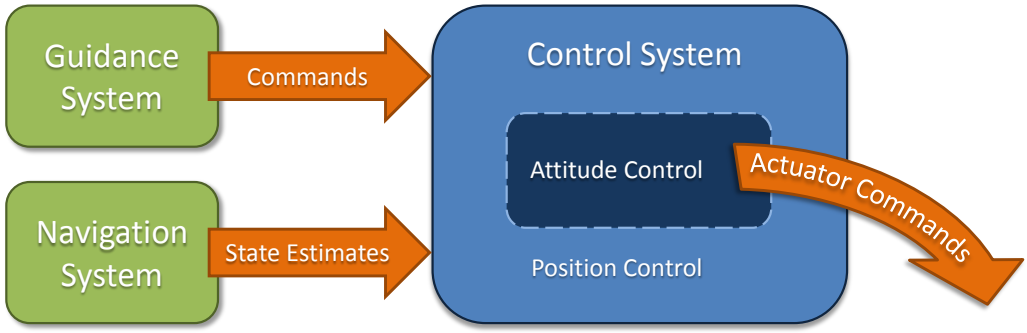


Figure 2: A common control architecture for unstable rotorcraft utilizes a set of nested loops to control attitude, velocity, and ultimately position by generating actuators commands. The guidance system provides position, velocity, and attitude commands, and the navigation system provides an estimate of the system states.

control system is not properly tuned. Even for stable flying vehicles, coupling between lateral and longitudinal motion, as well as aerodynamic interaction with the environment, must be considered. Fortunately, much progress has been made in the area of designing advanced control systems for rotorcraft vehicles [14]. Typically, a position control loop generates a velocity command, a velocity control loop generates an attitude command, and an attitude control loop generates servo commands to stabilize the vehicle by controlling the angular rate [27, 30, 12, 56].

This system of nested control loops (see Figure 2) requires that the vehicle maintain an estimate of its position, velocity, attitude, and angular rate, with the addition of an external inertial position reference if position control is to be ultimately achieved. The navigation system described in Chapter 4 relies on this type of control architecture, which utilizes existing software developed at the Georgia Tech UAV Research Facility [27]. For this research, contributions primarily lie in the integration of the laser scanner position updates with the existing navigation filter to provide better state estimates, rather than in advancements in the control system. For the stable air vehicle described in Chapter 3, however, new control algorithms were required. Due to the simplicity of the vehicle and the type of control required, traditional classical control methods utilizing independent control loops were used, with gain-scheduling techniques added to handle special cases.

2.2 The Extended Kalman Filter

This section, summarized from [21], describes how a linear dynamical system can be expressed in state space form, where the system states evolve according to the model shown in Equation 1. The rate of change of the states is a linear function of the states, $\mathbf{x}(t)$, and the inputs, $\mathbf{u}(t)$, plus noise, $\mathbf{w}(t)$. It is assumed that measurements of the states, $\mathbf{z}(t)$, are also a linear function of the states, plus noise, $\mathbf{v}(t)$, as shown in Equation 2.

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) + \mathbf{w}(t) \tag{1}$$

$$\mathbf{z}(t) = \mathbf{H}(t)\mathbf{x}(t) + \mathbf{v}(t) \tag{2}$$

Here, $\mathbf{H}(t)$ is the measurement matrix relating the measurements to the states. The vectors $\mathbf{w}(t)$ and $\mathbf{v}(t)$ are assumed to be uncorrelated zero mean Gaussian noise with covariance matrices $\mathbf{Q}(t)$ and $\mathbf{R}(t)$, respectively. A familiar form of the continuous Kalman filter (or Kalman-Bucy filter) used to estimate the states of such a system consists of two differential equations as shown in Equations 3-4:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{A}(t)\hat{\mathbf{x}}(t) + \mathbf{B}(t)\mathbf{u}(t) + \mathbf{K}(t)\big(\mathbf{z}(t) - \mathbf{H}(t)\hat{\mathbf{x}}(t)\big) \tag{3}$$

$$\dot{\mathbf{P}}(t) = \mathbf{A}(t)\mathbf{P}(t) + \mathbf{P}(t)\mathbf{A}^T(t) + \mathbf{Q}(t) - \mathbf{K}(t)\mathbf{R}(t)\mathbf{K}^T(t) \tag{4}$$

where $\dot{\hat{\mathbf{x}}}(t)$ is the rate of change of the state estimate, $\hat{\mathbf{x}}(t)$. The covariance of the estimate is given by $\mathbf{P}(t)$, and it is a measure of the uncertainty of the estimate at the current time. The Kalman gain, which minimizes the mean-square error of the measurements with respect to the expected measurements, is given by:

$$\mathbf{K}(t) = \mathbf{P}(t)\mathbf{H}^T(t)\mathbf{R}^{-1}(t) \tag{5}$$

For most real systems, flying vehicles in particular, the state vector changes as a nonlinear function of the states and inputs at a given time. Likewise, measurements may also be a nonlinear function of the states and inputs. In practical systems, measurements are typically collected at specific time increments, as most sensors have either a fixed update rate or are sampled at a fixed rate. Equations 6-7 shows the model and measurement equations for the general nonlinear case of a continuous dynamic model with measurements at discrete times, t_k [21].

$$\dot{\mathbf{x}}(t) = f\big(\mathbf{x}(t), \mathbf{u}(t), t\big) + \mathbf{w}(t) \tag{6}$$

$$\mathbf{z}_k = h_k\big(\mathbf{x}(t_k), \mathbf{u}(t_k), t_k\big) + \mathbf{v}_k \quad ; \qquad k = 1, 2, 3, \ldots \tag{7}$$

While the standard Kalman filter requires linear system dynamics, the nonlinear system can be linearized about the instant in time when measurements are taken. This is the essence of the Extended Kalman Filter. Equations 8-9 show the propagation equations for the continuous state estimate and covariance matrix [21]:

$$\dot{\hat{\mathbf{x}}}(t) = f\big(\hat{\mathbf{x}}(t), \mathbf{u}(t), t\big) \tag{8}$$

$$\dot{\mathbf{P}}(t) = \mathbf{F}\big(\hat{\mathbf{x}}(t), \mathbf{u}(t), t\big)\mathbf{P}(t) + \mathbf{P}(t)\mathbf{F}^T\big(\hat{\mathbf{x}}(t), \mathbf{u}(t), t\big) + \mathbf{Q}(t) \tag{9}$$

Equations 10-11 show the update equations [21], whereby sensor measurements are used to correct the state estimate and the covariance at time t_k. A superscript notation is used to represent the state estimate at time t_k just before the update $\hat{\mathbf{x}}_k^-$ and just after the update $\hat{\mathbf{x}}_k^+$, with the same notation used for the covariance matrix. The Kalman gain matrix is given by Equation 12 [21].

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\big(\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-)\big) \tag{10}$$

$$\mathbf{P}_k^+ = \big(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k(\hat{\mathbf{x}}_k^-)\big)\mathbf{P}_k^- \tag{11}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\hat{\mathbf{x}}_k^-)\big(\mathbf{H}_k(\hat{\mathbf{x}}_k^-)\mathbf{P}_k^-\mathbf{H}_k^T(\hat{\mathbf{x}}_k^-) + \mathbf{R}_k\big)^{-1} \tag{12}$$

The linearization of the state propagation and measurement equations is accomplished by forming the Jacobian matrices $\mathbf{F}\big(\hat{\mathbf{x}}(t), \mathbf{u}(t), t\big)$ and $\mathbf{H}_k(\hat{\mathbf{x}}_k^-)$ as shown in Equations 13-14 [21].

$$\mathbf{F}\big(\hat{\mathbf{x}}(t), \mathbf{u}(t), t\big) = \left.\frac{\partial f\big(\hat{\mathbf{x}}(t), \mathbf{u}(t), t\big)}{\partial \mathbf{x}(t)}\right|_{\mathbf{x}(t)=\hat{\mathbf{x}}(t)} \tag{13}$$

$$\mathbf{H}_k(\hat{\mathbf{x}}_k^-) = \left.\frac{\partial h_k\big(\hat{\mathbf{x}}(t_k), \mathbf{u}(t_k), t_k\big)}{\partial \mathbf{x}(t_k)}\right|_{\mathbf{x}(t_k)=\hat{\mathbf{x}}_k^-} \tag{14}$$

In this research, an existing EKF-based navigation filter developed at the Georgia Tech UAV Research Facility is utilized as part of the indoor navigation system for an unstable rotorcraft vehicle. The existing navigation filter is designed to operate using GPS to provide position updates. As described in Chapter 4, 2-D position and heading are measured using a laser scanner, and altitude is measured using a sonar. These measurements, with their respective errors, are used to update the EKF in addition to the IMU measurements. A brief description of how the laser scanner is used to provide localization is provided below, with specific implementation details provided in Chapter 4.

2.3 Simultaneous Localization and Mapping (SLAM)

For Simultaneous Localization and Mapping, much emphasis is placed on retaining a history of scan data and associated pose information, with the eventual goal of using pose constraints to find an optimal solution of past motion. Thus, a more accurate estimate of past motion and present state is achieved, improving not only knowledge of the vehicle's location and orientation, but also improving the accuracy of the map by correcting past observations when a pose constraint chain is solved. Solving this optimization problem comes at a computational price, however, especially when using a nonlinear solver.

For GNC algorithms that only require an estimate of very recent history, solving the pose constraint chain and correcting past motion estimates is less important and may not be

worth the computational effort. The EKF navigation solution maintains a current estimate of vehicle pose and motion, using a recent history of sensor measurements. Error in the measurements, as well as plant model, cause this solution to drift over time. Throughout this thesis, the concept of local versus global error is important to the development of the navigation systems discussed herein. In this thesis, *global error* is defined as the total accumulated error between the vehicle's current inertial position estimate and its initial position. *Local error* refers to the error in the position estimate relative to the local environment directly detectible by the vehicle's range sensors. Global error causes distortions in maps generated by the system, often causing spatial compression in certain directions, or bending of hallways that are straight in reality. Local error results in a diminished capability for the vehicle to maintain position control relative to immediately observable features in the environment (distance from a wall, for example). If the position error relative to the current local environment is small, a vehicle can be stabilized and flown indoors even though global error with respect to the initial state is large. A global map generated from observations throughout the flight will accumulate large error if corrections to past state estimates are not made. However, depending on a vehicle's mission and its associated guidance algorithm, global map errors may not affect mission performance.

The underlying basis for this research is that a variety of missions requiring indoor flight are not negatively impacted by global position error. As depicted in Figure 1(a), an air vehicle may not require a map at all. Even when mapping is necessary to complete the mission, some level of distortion in the map may be acceptable. If the vehicle does not require a globally consistent map for its navigation, then the effect of map errors is primarily dependent on how the map is used by ground observers. The primary GNC task for the aerial vehicle is to provide attitude stabilization, bound position drift, and avoid collision with the environment while maximizing mission goals such as area explored, observations made, number of targets identified, or other similar objectives. Particularly for indoor aerial navigation, smaller vehicles and shorter flight times limit computational power and geographic scope of the mapping and localization problem. As a result, the SLAM methods tailored specifically for indoor flight during this research place an emphasis on simplicity and

speed of computation, with tolerable accuracy, over algorithms that produce a very accurate global map. Due to the unique requirements of indoor aerial navigation, it is difficult to compare the performance of SLAM algorithms discussed here with those developed for ground vehicles with comparatively unlimited computational power and vehicle dynamics that are slow, stable, and highly predictable.

Most existing SLAM algorithms can be classified by their propagation method as either based on the Extended Kalman Filter, a Particle Filter, or some hybrid combination of the two. In either case, the problem essentially boils down to making observations of features in the environment from different vantage points which facilitate the estimation of the vehicle's position and attitude (together referred to as the vehicle's pose). Pose estimation is greatly improved by using some method to measure the vehicle's motion, which can then be used to predict the vehicle's pose before making corrections via environmental observations. Odometry can take many forms, from simply measuring wheel rotation on ground vehicles, to the use of a complete state estimation algorithm that incorporates measurements from inertial sensors, magnetic, and air data sensors on flying vehicles. Measurement error in the odometry and uncertainty in the plant dynamics contribute to uncertainty in the vehicle pose estimation. It is the task of the SLAM algorithm to use observations of features in the environment, which have their own associated uncertainty, to reduce the uncertainty of the vehicle pose estimate as much as possible. Algorithms based on the EKF and the Particle Filter apply knowledge of observed features slightly differently.

2.3.1 EKF SLAM

In EKF SLAM, landmarks are observed and the pose associated with each is stored along with the vehicle pose in a state vector. This subsection summarizes the work described in [16, 17, 3], whereby an EKF is used to estimate vehicle pose and landmark position using vehicle odometry and observations of the environment. As new landmarks are observed, more states are added to the state vector. Vehicle dynamics are modeled in the filter, and as temporal and measurement updates are made the vehicle pose estimate is improved. This process takes advantage of the fact that although landmark observations from different

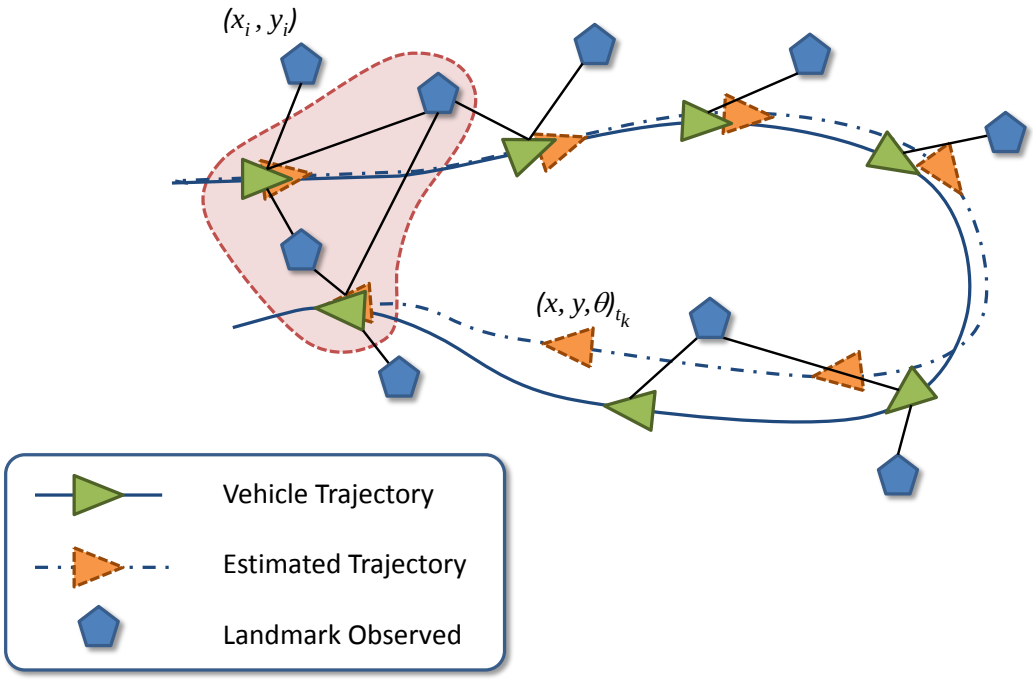


Figure 3: In EKF SLAM, the position of each landmark observed (x_i,y_i) is stored in a state vector, along with the vehicle state (x,y,θ). The observations are uncorrelated, however the *error* in the observations is highly correlated. The uncertainty of the landmark positions and vehicle state grows as the vehicle moves, but subsequent reobservation of landmarks allows the vehicle motion history to be estimated more accurately and the vehicle state estimate can be corrected.

vantage points are not correlated, the *error* in landmark observations is highly correlated. Thus, multiple observation of the same landmarks improves the vehicle pose estimate. One drawback to EKF SLAM is that the state vector and associated covariance matrix increases with each new observation. Managing these large matrices becomes a computational burden. As a result, much work has gone into ways of reducing that burden, including removing landmarks that are no longer needed and finding rapid inversion techniques for large sparse matrices. Even using such techniques, EKF SLAM is often considered impractical when observations come in the form of extremely rich data sets such as those generated by laser scanners or machine vision cameras. These sensors may produce observations of hundreds of landmarks or features at a very high rate, such that the associated state vector soon becomes unmanageable for real time applications.

2.3.2 Particle Filtering SLAM

As a method for taking advantage of the large quantity of data produced by scanning and imaging sensors, Particle Filters (PF) are an improvement over EKF SLAM. An overview of PF SLAM can be found in [17], with more details available in [64]. In a particle filter, multiple pose estimates, referred to as "particles", are generated and propagated forward using a dynamics model (which may be as simple as a Newtonian integration of the vehicle's estimated velocity). Each particle is perturbed based on the uncertainty of the pose estimate, such that the group of particles represents a growing cloud of probable vehicle poses. Observations in the form of laser scans or images are used by comparing changes in the observations from one data set to the next, allowing vehicle motion to be estimated. Thus, changes in vehicle pose from one instant to the next can be observed and used to correct the pose estimate stored for each particle as it is propagated along. These observations form a chain of constraints on the vehicle pose over time. Eventually, all of the particles are evaluated based on these pose constraints, and the optimal solution is found that satisfies all of the constraints. The particle which has the motion history that best meets the constraints is considered to be the best estimate of vehicle pose, a series of new particles is spawned from that pose, and the process continues. The corrected vehicle motion history over the life of the particle is then used to correct past scan information for the purpose of reducing errors in the global map.

With both the EKF and Particle Filter approaches, uncertainty in the pose estimate continues to grow over time, although the rate is slower than purely integrating vehicle odometry due to the repeated cycle of observing-correcting. This error can be virtually eliminated on a global scale by returning to a previously observed location on the map and correctly determining the total pose error accumulated over the history of the vehicle's motion. Solving this "loop closure" problem can correct large errors in a vehicle's path estimate, though that path may be hundreds of meters long, if a complete history of odometry and observations is stored and the proper data associations are made in order to recognize that a vehicle has returned to a previously visited location. The approach to SLAM in this research could be described as particle filtering with only a single particle, since there is

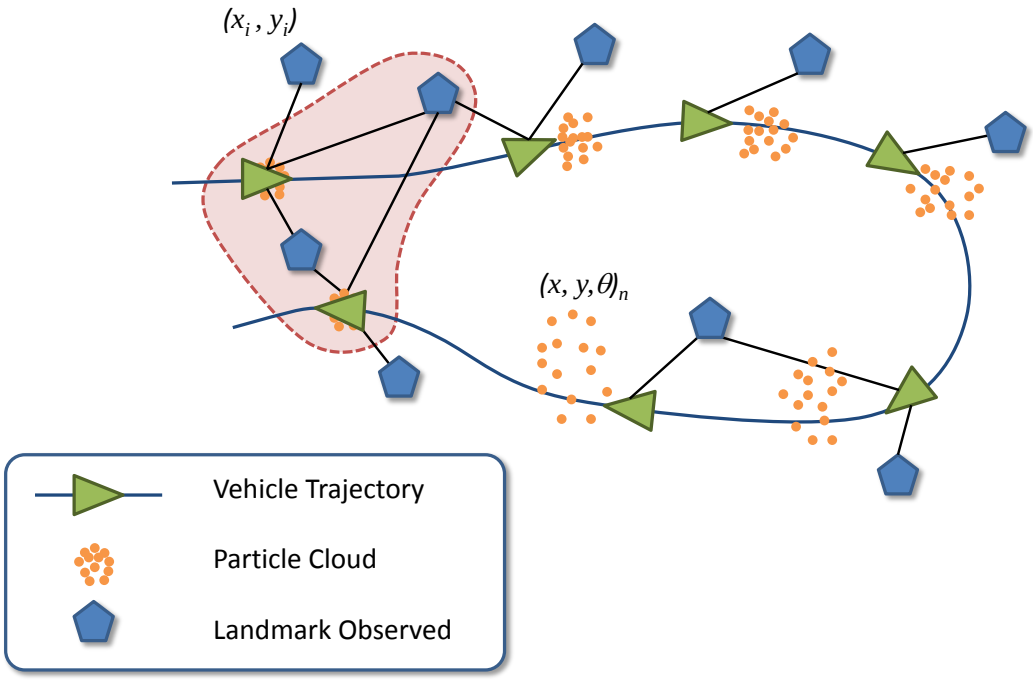


Figure 4: In Particle Filtering SLAM, the a cloud of particles, each with its own state estimate $(x,y,\theta)_n$, is propagated with uncertainty along the estimated trajectory. The cloud grows until subsequent reobservation of landmarks allows the historical pose constraint chain to be solved. The particle whose path history best solves the pose constraint chain is selected, and a new group of particles is spawned.

only one copy of the vehicle pose and its associated covariance. No loop closure is attempted and no corrections to the map are made, so having multiple hypotheses on the pose is not beneficial. The particle (pose estimate) is propagated forward using the vehicle's EKF state estimation, with pose measurements accomplished by analysis of the laser scan data.

2.3.3 Extracting Information from Laser Scans

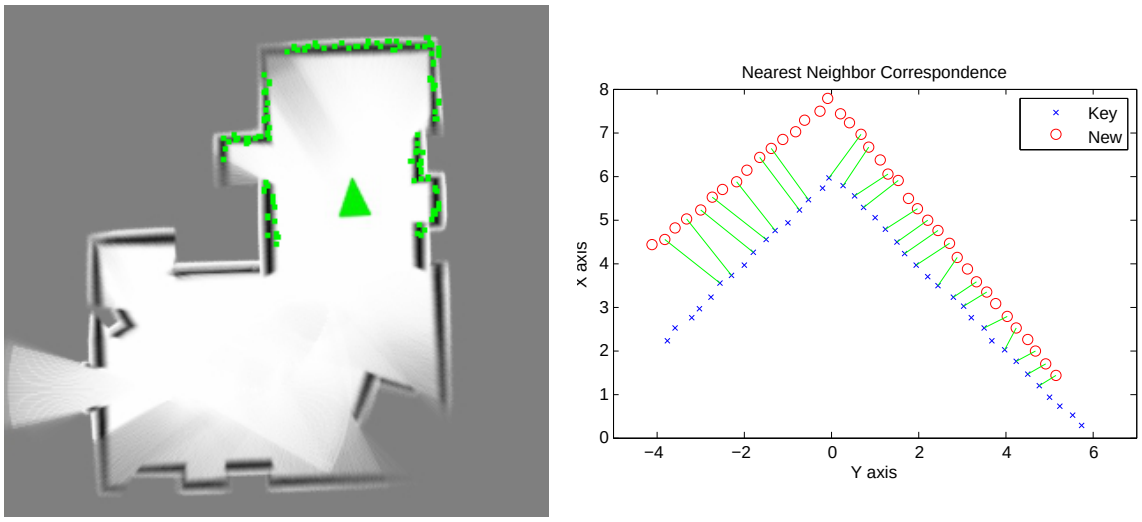
Scanning laser range finders produce a data set rich with information. Instead of a single measurement, they produce hundreds of range readings each scan cycle over a wide field of view. Typically, the data streams continuously and in order to make use of it any navigation system must be able to process it in real time. Range measurements are provided in sequence, beginning and ending in a known direction, with a known angular separation between each measurement. Field of view varies from sensor to sensor, usually somewhere

between 180° and 270° for most sensors. Angular resolution varies as well, though most sensors can deliver 1° separation or lower between range measurements. Sensor range is typically directly related to size, weight, and power consumption. Small, lightweight sensors are available with a range of 4m, while some larger sensors have a range up to 80m while still being portable. Most scanners that are suitable for indoor flight are in the 4m-30m range, weighing between 160g-400g typically.

Range measurement error is primarily a function of the range measured, incidence angle between the beam and the target, and to some degree the target material, although this usually affects the probability of a successful measurement more than the measurement error itself. Since target orientation and material properties are usually unknown, sensor noise is typically modeled as zero mean Gaussian white noise in the range measurement direction. Noise in the angle of measurement is commonly neglected [2, 43].

Information contained in a laser scan can be used in a variety of ways. Most common methods of analyzing scans can be classified as scan matching (one scan is compared with another scan), scan registration (a scan is compared against a set of stored values, such as a map) or feature extraction (salient features are identified through various methods). Each type of analysis has benefits and drawbacks, with trade-offs between accuracy, processing time, information content, ease of implementation, and type of information produced, among others. Scan data is usually supplied in polar coordinates, though conversion to cartesian coordinates at some point in the process is typical for most navigation applications, and different algorithms will make this conversion at different points. Scan matching and scan registration algorithms typically provide a level of confidence, indicating how well the algorithm is able to match a scan with the reference. This match confidence, however, does not provide a complete picture with regard to the uncertainty of the pose estimate. Certain environments provide few features, or features only in certain directions, upon which to accurately estimate vehicle pose. A novel approach to addressing this problem is presented later, which can be applied to both scan matching and scan registration algorithms.

Scan registration techniques require the creation and storage of a global map. The map represents the extent of the space explored by the robot, broken up into a grid. Map



(a) Scan registration

(b) Scan matching

Figure 5: In scan registration (a), scan measurements are compared to a map to determine the likely vehicle pose. Scan matching (b) algorithms compare one scan directly to another scan to estimate the change in pose.

information varies, from a simple binary representation of obstacles and free space, to a complete probability map where each grid square contains a number representing the likelihood that an obstacle exists there. Map resolution and total size affect speed and accuracy of the localization process. Clearly, a trade-off exists between the map fidelity and how quickly an algorithm can process and compare a scan relative to the map if an exhaustive search of the map is required. For global localization techniques, this is indeed the case, as detecting large loop closure and solving the "kidnapped robot" problem are often important goals. If the map search is restricted to local motion, using a Monte Carlo routine near the pose estimate for example, then the map resolution affects only required memory and not the speed of the algorithm. In either case, scan information is compared to values on the map for a given pose hypothesis, and some measure of how well the scan matches the map is calculated. Different pose hypotheses are evaluated, and the one that results in the best match is used to update the vehicle pose estimate.

In contrast to scan registration, scan matching does not require the use of a global map. This process involves comparing scan data from one scan directly to scan data collected previously. The baseline scan data may be a single scan or a set of previously stored scans. A common implementation is sequential scan matching, whereby each new scan is compared

to the previous scan as a means of vehicle odometry. The output of the scan match in this case is a change in vehicle pose, with associated uncertainty, which can be used to update the vehicle navigation estimate. Scan matching can be based directly on the distance between individual corresponding data points, differences between parameterized models of the two scans, or the distance between corresponding features extracted from the data sets. These features are typically line segments, polynomial curves, or corners identified in the scan. Feature extraction can be used for both scan matching and to provide input to the vehicle guidance system by identifying topology of the environment. Figure 5 shows examples of scan registration and scan matching for estimating the vehicle pose given a laser scan measurement.

Scan Registration, scan matching, and feature extraction are all problems that have been well studied in the computer science and robotics fields as cited in Chapter 1. However, due to the only recent emergence of aerial based SLAM experimentation, most algorithms favor robustness and accuracy over computational speed. As a result, choice of navigation algorithm should take into consideration suitability for implementation onboard a small flight-worthy computer. This research examines several algorithms and identifies a few of the faster methods for localization as a way of determining the minimum requirements for performing indoor flight navigation. Thus, the laser-aided navigation system described herein is designed and tailored specifically for small flying vehicles. Specific details on the algorithms and their implementation and testing are provided in Chapter 4.

CHAPTER III

LOW COST RELATIVE GUIDANCE, NAVIGATION, AND CONTROL

Successful indoor flight does not necessarily require the use of cutting-edge sensor technology, the fastest and smallest computer available, all mounted on an advanced, highly maneuverable vehicle. As evident by observing insect behavior, modest navigation and obstacle avoidance goals can be accomplished with a much simpler system. However, to avoid mimicking the proverbial moth fluttering around a lamp, some planning must go into the guidance system so that simple relative navigation can be used to effect productive behavior. To that end, this chapter discusses an approach to indoor navigation that relies on low-cost sensors, simple GNC algorithms running on a small microcontroller, flown on a passively stable vehicle. Simulation and flight test results are presented at the end of the chapter, describing the success of this approach to autonomous indoor flight.

3.1 Vehicle and Sensors

The designing a system for indoor flight requires the consideration of a variety of aircraft and sensors, with the GNC system largely influenced by the type of behaviors desired. The characteristics of the vehicle and the sensors carried onboard must be thoroughly understood in order to ensure the overall system is capable of performing the required mission. The following section provides a discussion of the vehicle and range sensors used to demonstrate simple autonomous indoor flight.

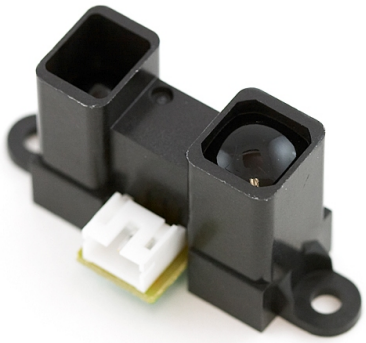
3.1.1 Aerial Platform

The choice of platform for indoor flight is influenced by several factors. Airspace is typically limited, so traditional fixed-wing platforms have a disadvantage. Most rotorcraft are unstable in flight and must have an inertial measurement unit (IMU) and some additional method to estimate velocity or attitude in order to be effectively stabilized. Single-rotor helicopters and quad-rotor designs fall into this category, where maintaining stable flight

Figure 6: The E-Sky® Big Lama. Photo courtesy E-Sky®. *Note: the tail rotor on this aircraft is neither functional nor required.*

without GPS or an external attitude estimation system is itself a challenge. For this reason, only stable platforms were considered for this low-cost approach to indoor navigation. Lighter-than-air craft were not considered feasible due to their limited payload capacity and poor maneuverability. The desire for a stable, commercially available vehicle with payload capacity resulted in the choice of a coaxial helicopter for the sensor platform. The coaxial helicopter has a pair of counter-rotating blades, making the vehicle more compact since no tail rotor is required for yaw control. The bottom set of blades has cyclic control for maneuvering, while the upper set of blades has a Bell stabilizer (sometimes called a flybar) to counteract vehicle pitch and roll, providing some attitude stability. Several manufacturers make radio-controlled coaxial helicopters of various sizes. The helicopter selected for this research was the E020 Big Lama [19], made by E-Sky® (see Figure 6). It has a rotor diameter of 46cm and has a mass of approximately 410g in its stock configuration. Initial flight tests indicated that the stock vehicle has a useful payload capacity of approximately 50g. Removal of the canopy and motor upgrades significantly increased the available payload capacity. The final flight configuration, with avionics, a 1350mAh lithium polymer battery, brushless motors, and a protective shroud had a total mass of 605g.

(a) MaxBotix® LV-MaxSonar®



(b) SHARP GP2Y0A02YK0F Infrared Sensor

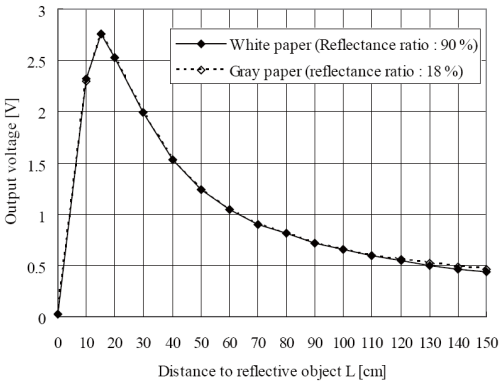
Figure 7: Lightweight, low-cost range sensors suitable for indoor navigation. Photos courtesy Sparkfun™ Electronics. [60]

Table 1: Range Sensor Manufacturer Specifications

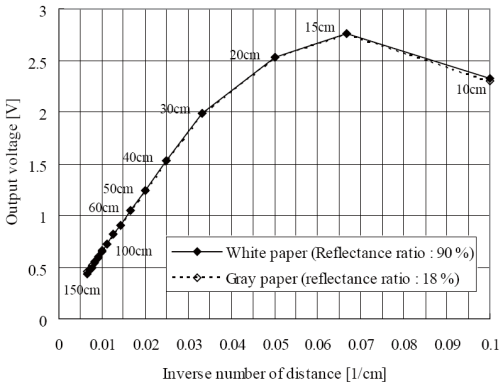
Sensor	MaxSonar® [41]	SHARP IR [55]
Range	0.15-6.45 m	0.2-1.5 m
Resolution	2.54 cm	1 cm
Weight	4.3 g	4.8 g

3.1.2 Sensor Selection

Several range sensors were selected based on their usable range and resolution, with an emphasis on low-cost commercially available sensors (see Figure 7. The MaxBotix® LV-MaxSonar®-EZ1 was chosen for measuring altitude. The SHARP GP2Y0A02YK0F infrared sensor was selected for close obstacle avoidance during forward flight and for wall-following navigation modes. It has a shorter range than the MaxSonar®, but it has a better resolution and a narrow beam width. In addition, multiple IR sensors can be operated at the same time with less interference than sonar. See Table 1 for detailed information on the sensors. The stock yaw rate gyro was replaced with a hobby heading-lock gyro to improve yaw stabilization.



(a) SHARP IR range/voltage response curve.



(b) SHARP IR inverse range response curve.

Figure 8: SHARP GP2Y0A02YK0F infrared range sensor response curves. [55]

3.1.3 Range Sensor Characterization

Both sensors were tested to determine the useful range and error characteristics. The SHARP infrared sensor has an analog output, and the correlation between output voltage and range is nonlinear. However, the reciprocal relationship is linear over most of the useful range of the sensor. Figure 8 shows the calibration curves provided by the manufacturer [55]. Actual voltage/range relationships were measured for two IR sensors and were comparable to the specification. In addition to determining the calibration curves, the sensors were each sampled for 10 seconds at different fixed ranges to determine the standard deviation. The analog output voltage is updated approximately every 38ms. The MaxBotix® sonar range finders output data via serial RS-232 protocol, a Pulse Width (PW) signal, or analog voltage output. For initial testing, the serial output was used. During flight tests, the PW signal was also used. For the PW signal, the sonar outputs a logical high signal that has a width of 147 microseconds per inch. Each measurement cycle for the sonar takes approximately 50ms. Operating multiple sonar at the same time can cause interference in some circumstances. However, the sonar can be operated in a chain, with each sensor triggering the next one automatically. Alternatively, the sonar can be triggered individually at specified times to avoid interference. The next sections provide further insight into the sensor characterization and data collection process for the IR and Sonar range sensors.

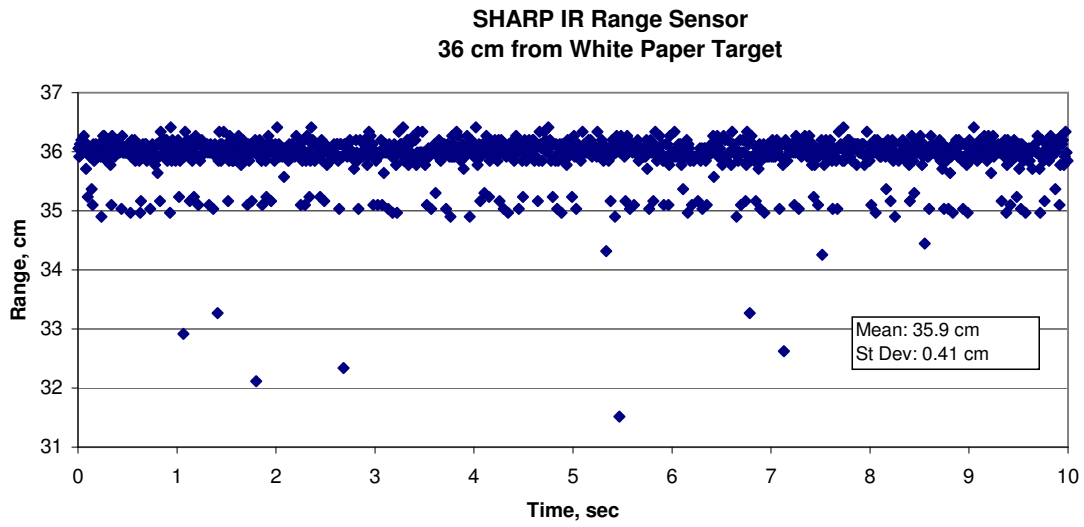


Figure 9: Initial test of the SHARP IR Range Sensor. The test was conducted at 36 cm from the target (white paper) for 10 seconds.

3.1.3.1 IR Sensor Testing and Calibration

Two identical IR sensors were tested to determine whether each sensor would require its own calibration curve. Initial testing consisted of collecting data from the range sensors at a fixed distance for approximately ten seconds. The analog output voltage was measured using a 10-bit analog-to-digital converter. The output voltage was then converted to range using the manufacturer's calibration curve. Figure 9 shows the result for a white paper target at a distance of 36cm. The data show about 90% of the measurements have a mean near the actual range, while approximately 10% are 1cm short of the actual range, with some sparse outliers reading as much as 4-5cm short. The sensor performed well, with a mean value of 35.91cm and a standard deviation of 0.41cm. However, the pattern observed in 10% of the data prompted a more in-depth analysis of the voltage output from the sensor.

The IR sensor was connected to an oscilloscope to determine the characteristics of the output voltage. As shown in Figure 10, the noise is periodic, consisting of a region approximately 0.13ms long with a period of 1.1ms. In this region, the output voltage is higher than the rest of the data, resulting in a shorter reported range. The time scale on the oscilloscope was reduced to record the output more accurately, and a voltage spike was observed at the

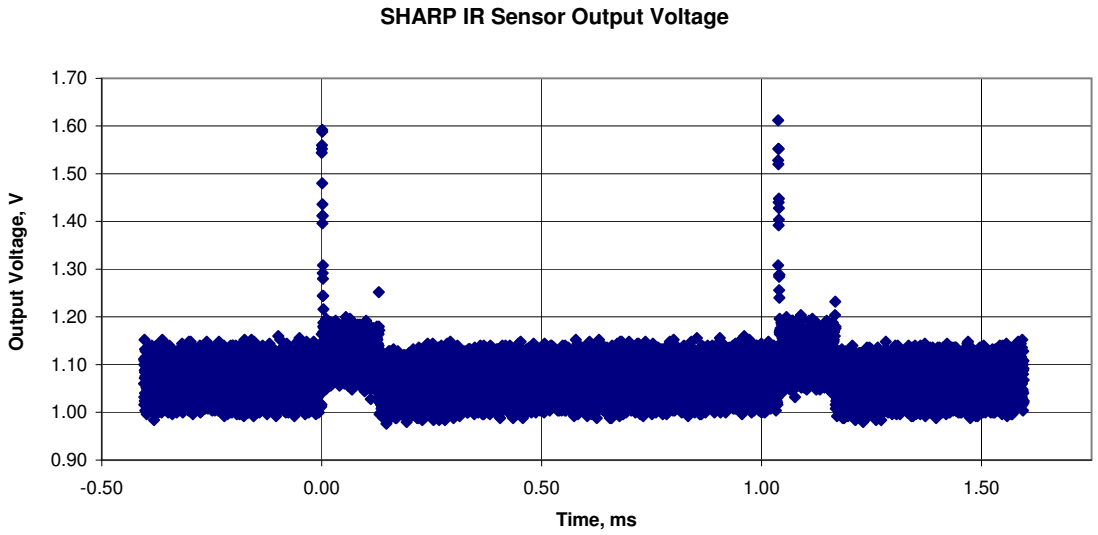


Figure 10: SHARP IR sensor output voltage. Note the periodic nature of the sensor noise.

beginning and end of the elevated reading (see Figure 11). It was determined that the characteristics of the noise fully account for the observed range measurement data. The 10% grouping of short range readings correlate to the elevated voltage output, while the outliers correlate to voltage recorded during one of the spikes seen at the beginning and end of the elevated region. As a result, a median filter was applied to the raw voltage, whereby three voltage measurements were collected for each range reading with the middle value of the three used to calculate the range. Using this filtering technique, the range measurement was greatly improved as shown in Figure 12. After filtering the data, the mean range reading was 36.00cm, with a standard deviation of 0.11cm. The three-point median filter described above was used on all subsequent IR measurements.

Next, the two IR sensors were tested at fixed intervals from a white paper target to determine the voltage-to-range relationship. The sensors were mounted to a tripod, and each sensor was operated independently to avoid interference. Then, output voltage was recorded for ten seconds each at 10 cm intervals from 20cm to 190cm. The output voltage was filtered before being recorded using the filtering algorithm described above. The data were averaged for the ten second sample, and the voltage for each sensor was plotted versus target distance. Figure 13 shows the voltage/range curve for the sensors tested. Although

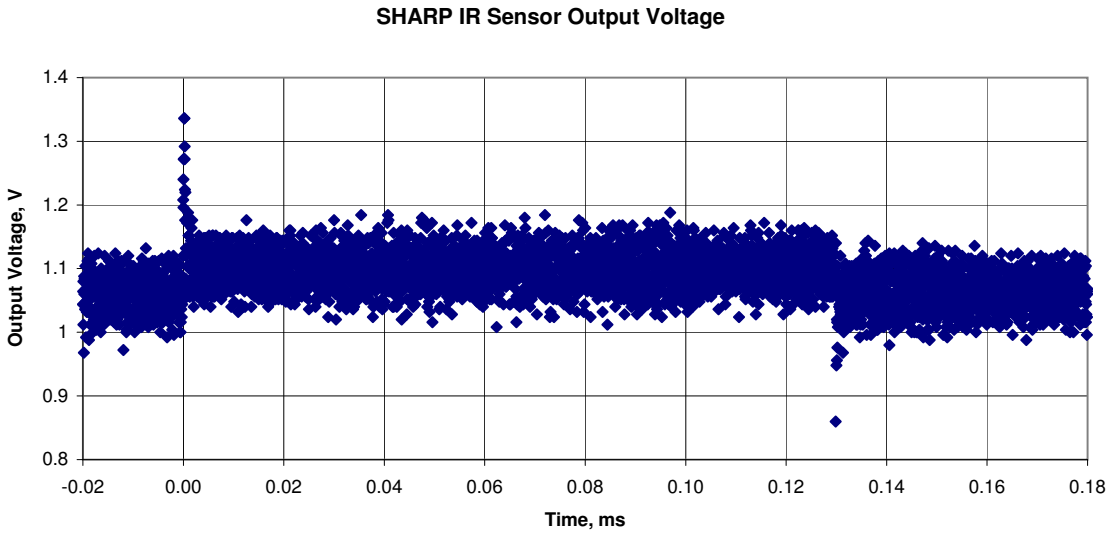


Figure 11: SHARP IR sensor output voltage, higher resolution capture. Note the spike before and after the elevated voltage region.

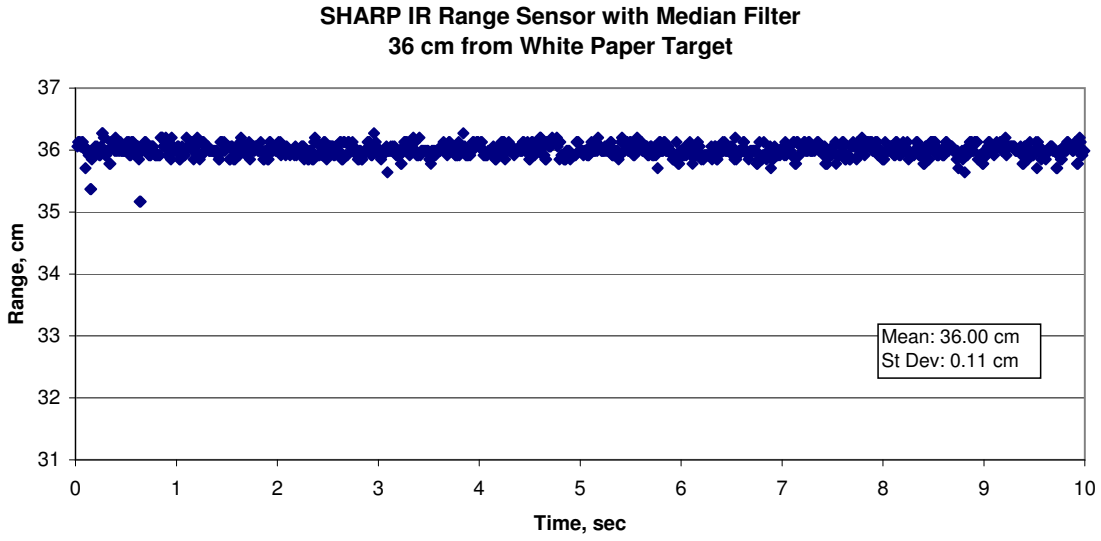


Figure 12: SHARP IR sensor output voltage using three-point median filter. Note the improvement in the mean and standard deviation.

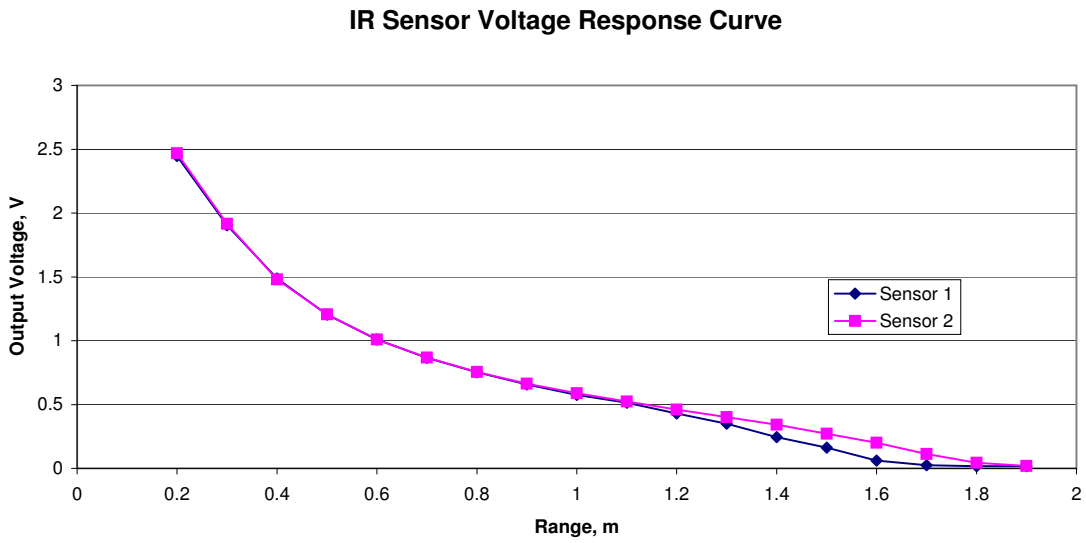


Figure 13: SHARP IR sensor voltage response curve. Each data point represents the average over a ten-second period, filtered as described above.

the sensor response is nonlinear, a graph of the voltage versus inverse distance shows some linear regions. Three linear regions were identified (see Figure 14) and linear approximations were used for these regions to calibrate the sensors. Using linear approximations for the relationship over these three regions produced an error of less than 6% over the manufacturer's advertised range of 20-150cm. The sensors may be used up to 190 cm under some circumstances, with calibration error of less than 10% (see Figure 15). The standard deviation was also calculated for the two sensors using data collected over the 10-second period at each distance interval (see Figure 16). Subsequent testing of a different batch of IR sensors showed that a single linear approximation can be used for the entire advertised range of the sensor.

After calibrating the SHARP IR sensors, they were both mounted to the test vehicle for initial flight testing. Due to mounting constraints, the initial test was done with the sensors mounted side-by-side. In practice, the sensors should be placed so that their illumination patterns are at least 20cm apart to avoid interference (the beam width of each sensor is approximately 10cm at maximum range). During the initial flight test, the vehicle was flown at varying altitude while data from the two sensors was recorded. Figure 17 shows the range

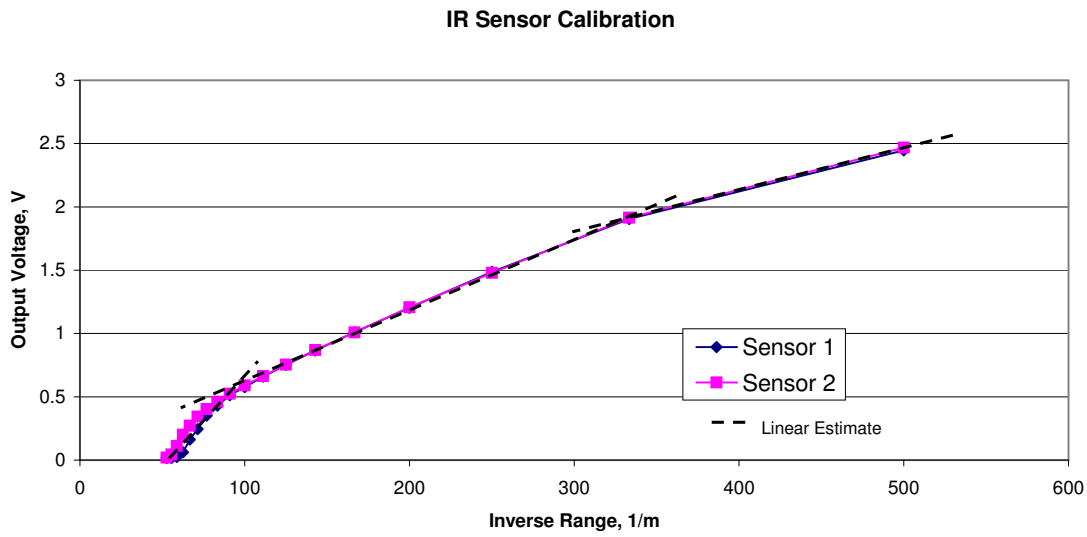


Figure 14: SHARP IR sensor calibration curves are nearly linear for the inverse range. The data were divided into three discrete linear regions.

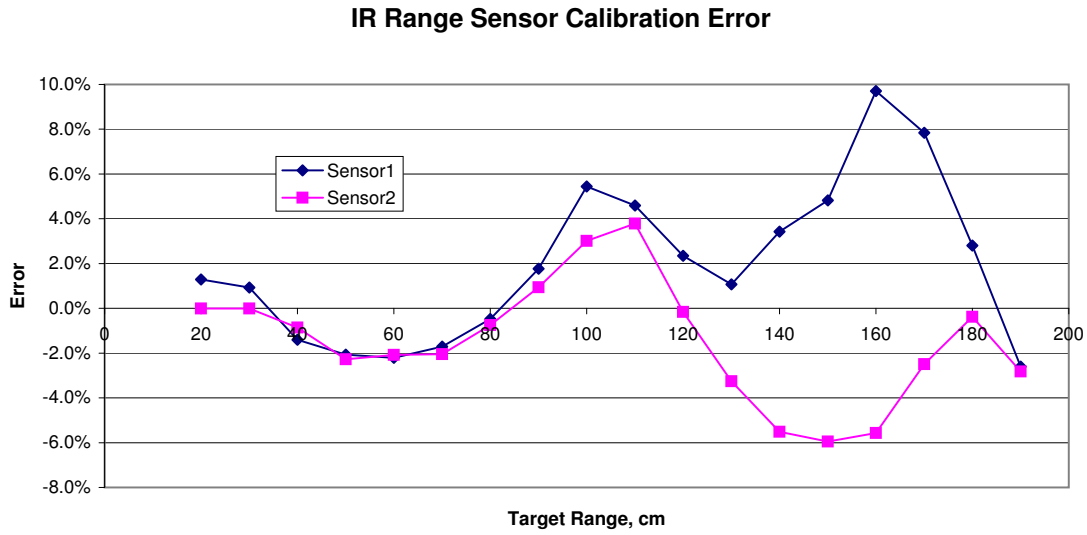


Figure 15: SHARP IR sensor calibration error. Three linear approximations are used for the voltage/inverse range relationship. Note: The manufacturer's advertised sensor measurement range is 20 - 150 cm.

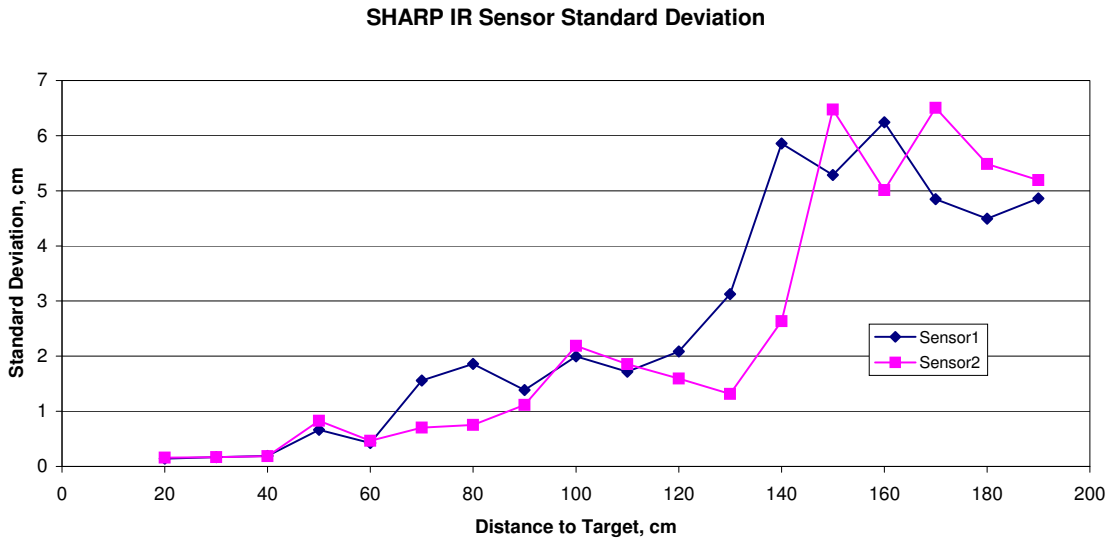


Figure 16: SHARP IR sensor standard deviation over measurement range. Two sensors were measured independently for ten seconds at each distance interval.

measurements recorded from both range sensors. Note that due to the nonlinear response of the sensor, distances below the minimum range appear higher than normal. For this test, range measurements after about eight seconds into the flight are valid. The two sensors performed nearly identically over most of their design range, however the measurements were quite noisy above approximately 120cm. This may have been due to interference between the sensors, which were operating simultaneously, since the standard deviation at this range was measured to be approximately 2cm for independent sensor operation.

3.1.3.2 Sonar Testing

Two ultrasonic range sensors were tested to determine which is better suited to indoor navigation. The MaxBotix® LV-MaxSonar®-EZ1™ and LV-MaxSonar®-EZ4™ sensors are both inexpensive and lightweight, making them possible candidates for small UAVs. The two sensors are nearly identical, but the EZ4™ has a narrower beam width than the EZ1™. Initial comparisons consisted of powering the sensors and reading the data via the RS-232 serial connection. The sensors were then aimed at targets at various ranges to determine qualitative difference between the two. Both sensors produce a digital output, and showed no noise when pointed at a fixed, flat target. In a cluttered environment, however, small

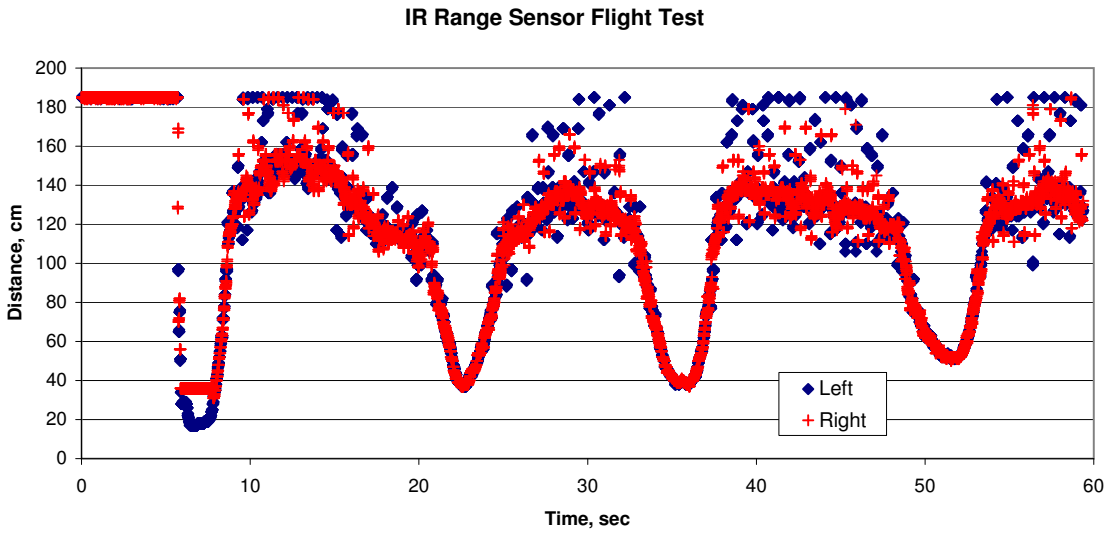


Figure 17: SHARP IR sensor flight test. Two sensors were placed side-by-side and aimed at the ground. During the flight, the vehicle was flown to different altitudes while recording data.

movements of the sensors produced "noisy" range readings due to varying strength of the sonar return from different objects. The sensors are designed to return the range to the first object detected. This can produce varying results in a cluttered, moving environment. The EZ4™ showed more noise in cluttered environments than the EZ1™, possibly due to its narrower beam hitting various objects of differing ranges when the sensor is in motion.

Once qualitative analysis was complete, the sensors were mounted to a tripod and tested in a more controlled environment. The tripod was set in the middle of a small room with the sensor platform 122cm above the ground, 173cm from the left and right walls, and 198cm from the front wall. The sensors were turned through a range of 180° beginning facing the right wall, turning counterclockwise to face the front wall, and finishing with the left wall. Each test was completed independently to prevent possible interference between the sensors. The sensors were turned by hand, though an attempt was made to ensure a fairly constant angular rate. Next, the sensors were operated simultaneously with similar results.

The range measurements of the initial test is shown in Figure 18. The results, similar for both sensors, illustrate a primary difference between a pulse-wave range measurement sensor

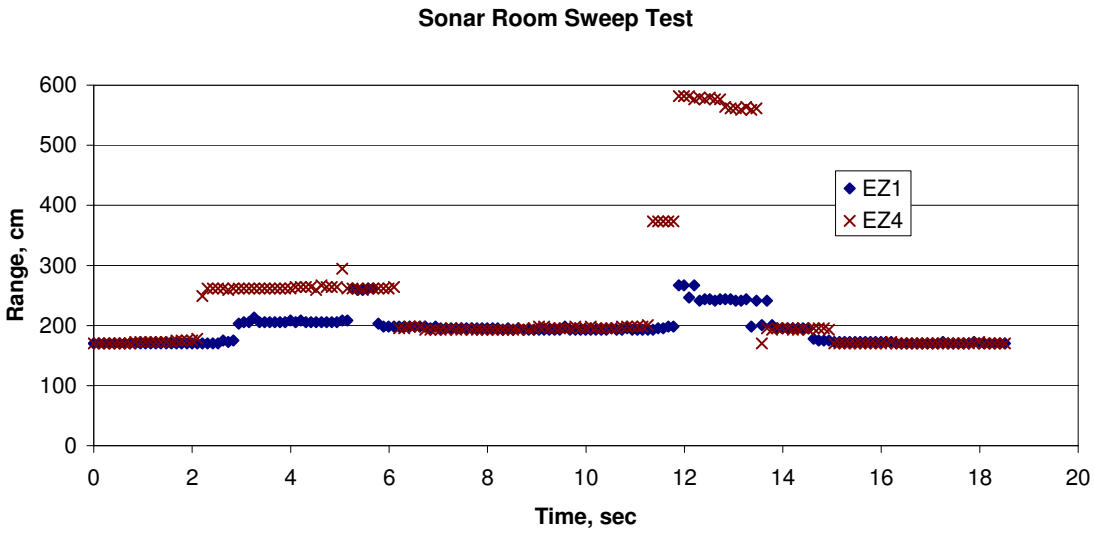


Figure 18: Distance measurements recorded using the EZ1™ and EZ4™ in simultaneous operation during a 180° sweep of a small room. Note the straight lines observed that correlate to the distance to the right, front, and left walls at 173cm, 198cm, and 173cm respectively. Some features such as corners and an open doorway are also observed.

(such as sonar or radar) and a point range measurement sensor (such as the IR sensors or a laser rangefinder). When facing a flat wall, the sonar measures the perpendicular distance to the wall, regardless of sensor orientation. The result is a reading which remains constant, until the sensor is turned past an angle where the wall is no longer detected. A graph of the range readings versus angle show a series of straight lines, with some features such as corners and doorways observed. The horizontal lines correspond to the perpendicular distance from the sensor to major feature points in the room. The first feature detected during the sensor sweep was the right wall at 173cm. As the angle increased, the sensors began to lose range measurement to the right wall and pick up the front right corner at 264cm. The EZ4™ sensor picked up the corner first, while the EZ1™ sensor with the wider beam maintained a reading to the right wall longer before picking up the corner. As the sensor angle was increased, both sensors eventually picked up the front wall, which was at a distance of 198cm. The reading remained fairly constant until the sensors were turned far enough to pick up an open doorway in the front left corner. As the sensors were rotated further, they both eventually picked up the left wall at 173cm.

Although the sensor orientation was not measured directly, the sensor platform was turned at a reasonably constant angular rate, and the time for each data point was recorded. As a result, the range data can be plotted on a 2-D map using the estimated angle information (see Figure 19). Although the sonar returns do not map directly to the wall locations, some features can be identified from the 180° sensor sweep. The arc that is visible from major geographic features represents the sonar wave pulse as it expands from the sensor. The sensor reports the distance of the first sonar return, which corresponds to the perpendicular distance to the walls and corners. Thus, the center of each visible arc in the 2-D plot represents the location of the sensor within the room. Given room dimensions and accurate heading information, it may be possible to localize a sonar sensor platform in a given room.

Further testing showed that the sonar range sensors return the shortest distance to any object or wall detected, regardless of the angle between the sensor and the object. As a result, small changes in sensor angle do not affect the range returned. For this reason, the sonar range sensor is an excellent choice for measuring the altitude of an indoor vehicle. Small changes in the pitch or roll angle would cause a downward-pointing line-of-sight sensor (like the IR sensor) to read long, whereas the sonar is unaffected. For this reason, the MaxBotix® LV-MaxSonar® -EZ1™ was chosen for altitude measurement. The sensor characterization tests demonstrate that the sonar have uniquely different properties than the IR sensors. The combined properties of the IR and sonar range sensors work together to improve overall vehicle navigation.

3.1.4 Sensor Integration

An ATMega128 microprocessor is used onboard the vehicle to read the sensors and process the data for navigation. The data are collected and filtered appropriately according to their error characteristics for altitude hold, simple obstacle avoidance and wall-following behavior. The IR range sensors have analog voltage outputs, which are read by onboard analog to digital input channels. One sonar, used for altitude, is read via serial port. The ATMega128 has one additional serial port, which is used for a data link. In addition to the sonar,

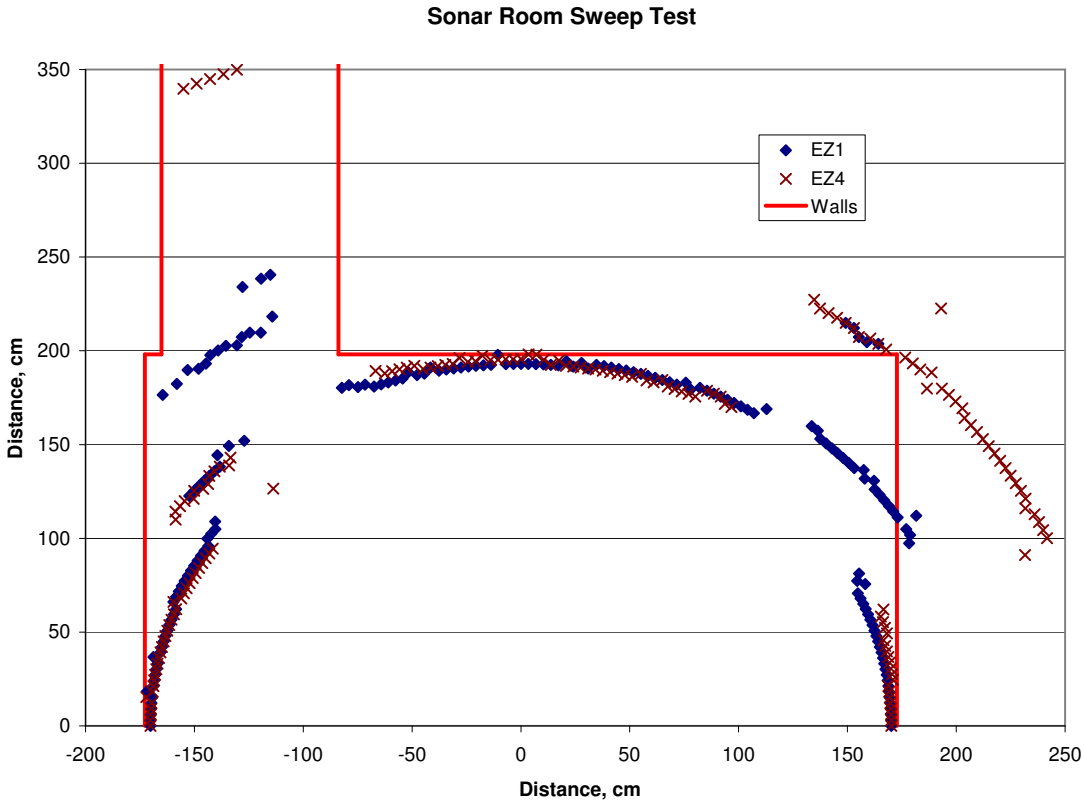


Figure 19: Distance measurements plotted using estimated sensor angle and measured range. Actual wall locations are show for reference. Note the EZ4™ more easily detects corners and open doorways than the EZ1™ .

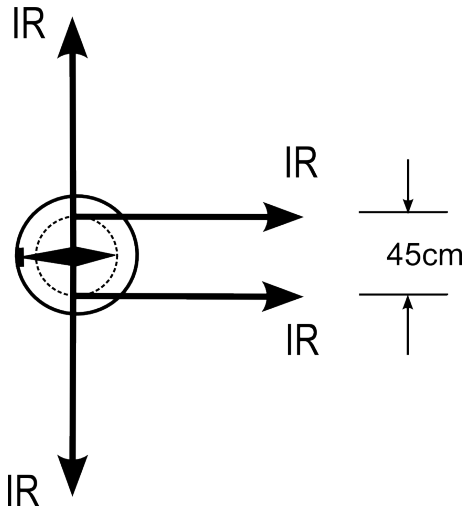


Figure 20: Range sensor layout as viewed from above with the vehicle facing right. The infrared sensors are placed and oriented as shown, while the sonar is pointed downward for sensing altitude.

two infrared sensors are placed 45cm apart looking forward for heading control, improved obstacle avoidance during forward motion, for detection of openings such as windows or doors, and for wall-following behavior. Two IR sensors are also used to sense obstacles in the lateral directions. Figure 20 shows the IR sensor configuration, and Figure 21 shows the sensors mounted to the test vehicle. Development of a custom carbon-fiber safety shroud enabled the vehicle to survive mild contact with obstacles, so a rear-facing IR sensor was not required.

3.2 Guidance Algorithm

A variety of guidance algorithms are possible using the sensor configuration described above. Possible behaviors range from simple random flight to thorough room-by-room exploration of an unknown indoor environment. This investigation into simple low-cost navigation was conducted in preparation for the 2009 International Aerial Robotics Competition (IARC) [26]. This competition, held in July 2009, required the autonomous entry into a building mock-up and successful navigation within the building in search of a specified target. Thus, the guidance algorithm flight tested and presented below is tailored to the simplest logic required to achieve IARC mission requirements. Using this simple guidance algorithm allows a simplified navigation scheme as well, whereby the vehicle only needs to

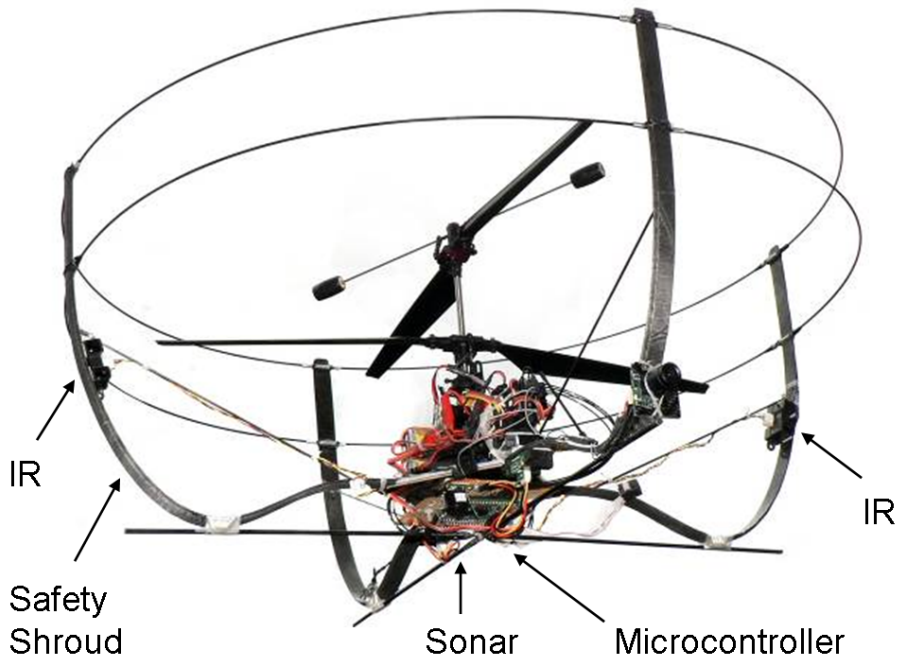


Figure 21: Flight vehicle with sensors and protective shroud installed.

determine its altitude, distance to one wall, heading with respect to that wall, and detection of obstacles in the vehicle's flight path. To accomplish those goals, an altitude-hold controller was used to maintain a fixed altitude throughout the flight, and a wall-following routine was used to explore the competition arena. Once in autonomous mode, the vehicle followed the guidance algorithm described below in Figure 22.

When the autopilot is enabled, the vehicle begins in "Window Entry" mode. In this

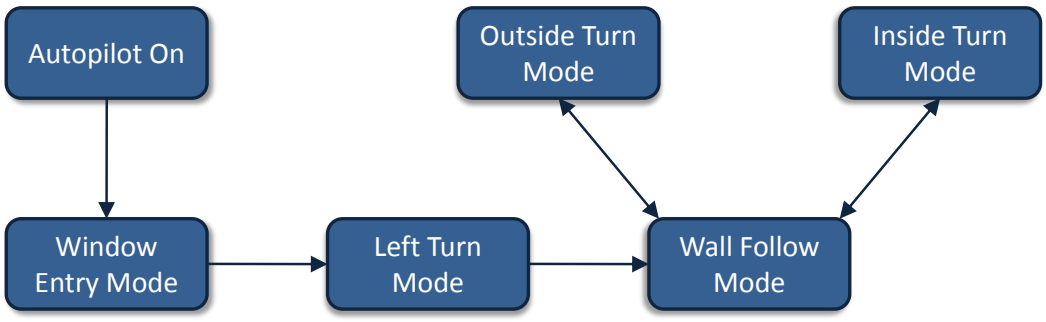
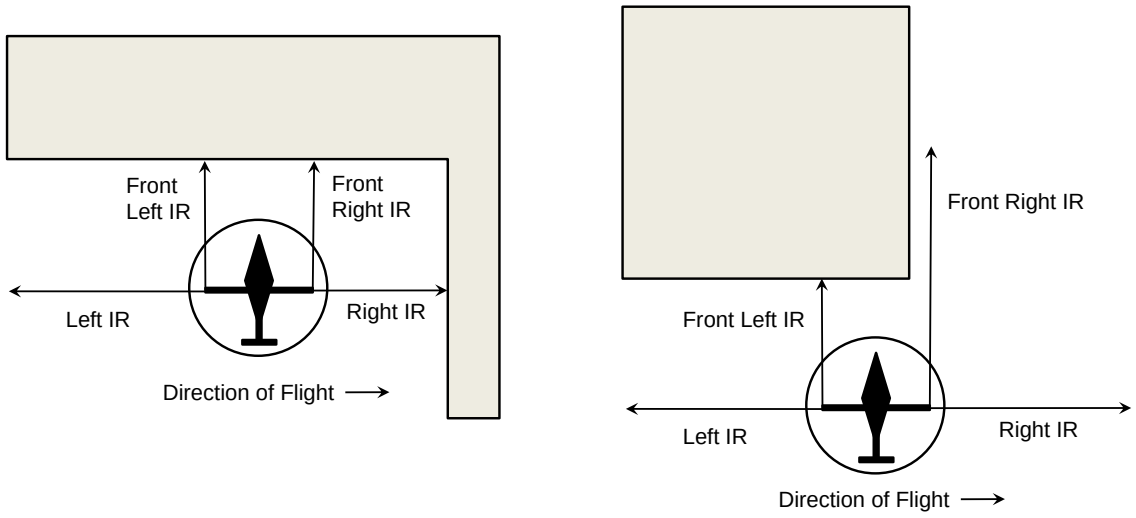


Figure 22: Wall-following guidance algorithm. Specific sensor inputs will cause the algorithm to progress to successive logic blocks.



(a) Inside turn

(b) Outside turn

Figure 23: When the vehicle is flying laterally, a wall detected in the direction of flight triggers the guidance state to enter an inside turn as shown in (a). If a the forward-looking range sensors detect an outside corner as shown in (b), the guidance system state switches to outside turn mode.

mode, it flies forward searching for the arena entry portal. If an object is detected by the forward-looking left or right IR sensors, the lateral controller adjusts the flight path so that the vehicle is in the center of the window. Once the vehicle enters the building, walls are detected by the left IR sensor and the vehicle enters "Left Turn" mode. In this mode, it turns to the left until the forward-looking IR sensors detect the wall. Once the forward-looking sensors detect the wall, "Wall Follow" mode begins. In this mode, the longitudinal controller maintains a commanded distance from the wall, while the heading controller maintains the desired heading with respect to the wall (navigation details are provided in the sections below). The vehicle then flies along the wall to the right, using the right facing sensor to detect walls and obstacles in the flight path. During lateral flight, the lateral controller does not try to maintain a fixed position, rather it monitors the side-looking sensors and tries to prevent the vehicle from getting within a specified distance of obstacles. As shown in Figure 23, different corner-turning modes are entered depending upon different conditions detected by the IR sensors. If a wall or obstacle is detected in the direction of flight, the vehicle enters "Inside Turn" mode, whereby it changes its heading to either turn the corner (for concave corners) or fly around the obstacle. This is achieved by giving an

open-loop yaw command until no obstacle is seen by the right IR sensor. Alternatively, if one of the forward-looking IR sensors detects a step increase in the range while the other sensor still reads near the estimated wall distance, a convex, or outside, corner has been detected. The vehicle then enters "Outside Turn" mode and the controller commands an open-loop yaw to the left in order to continue around the corner. Once an inside or outside corner has been completed and valid range measurements are seen on the two front IR sensors, the vehicle returns to "Wall Follow" mode and continues flight. Using this simple event-based guidance system, it is possible for the vehicle to effectively traverse the majority of an unknown indoor environment. Simple variations, such as random changes in flight direction, could improve coverage of areas not encountered by the basic algorithm. This simple guidance system relies only on range measurements relative to the local environment, which requires only a simple relative navigation algorithm.

3.3 Navigation Algorithm

The guidance system described above does not require any inertial navigation or other global position estimate. As a result, the navigation algorithm is much simpler than those utilized on most UAV systems. Only five range measurements are required, all relative to the local environment. Different filtering methods are used to maintain current estimates of altitude, heading, and distance to walls. The sensor characterization experiments described above revealed that both the IR and sonar range sensors occasionally reported readings near maximum range when the sensors failed to detect a wall or obstacle at a particular instant. These and other outliers can cause catastrophic results for a feedback control system that expects smooth error measurements. For this reason, a smart filtering routine described in [13] was implemented to prevent large step or impulse inputs from adversely affecting the controller. In this routine, a Kalman filter is used to estimate the range and range rate for each of the sensors. The covariance of the residuals is then used to detect and ignore outliers in the range measurement beyond three standard deviations. Occasionally an actual discontinuity in range occurs, such as flight over an obstacle on the ground. The smart filtering routine recognizes such events when a number of similar measurements is detected

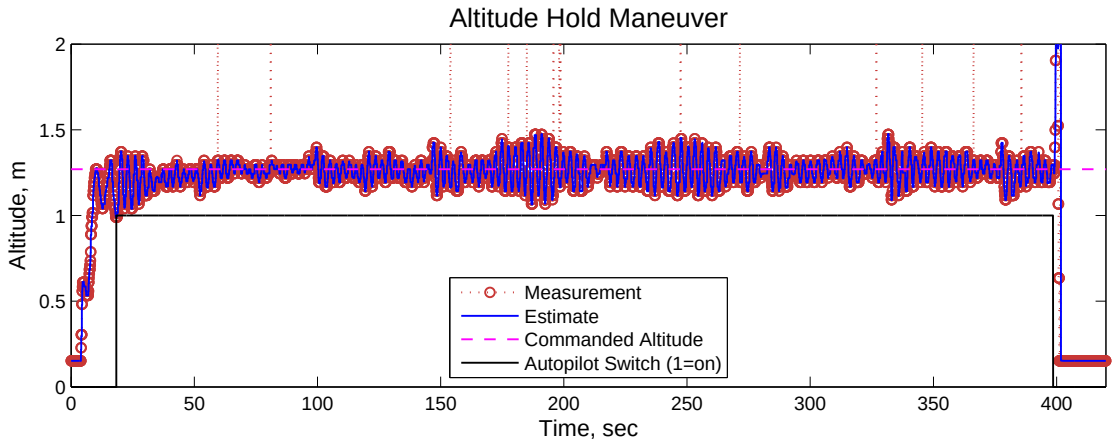


Figure 24: Altitude measurements and estimate. The performance of the outlier detection routine is visible here. Outliers typically have values near the minimum range (0.15m) or the maximum range (6.45m). The autopilot command represents the enabling and disabling of the altitude-hold controller.

in sequence, and the filter adjusts the range estimate to match the new measurements without changing the velocity estimate. The result is a smooth vehicle response to new range information even in the presence of step changes in range measurements. Figure 24 shows an example of the smart filter in operation during an altitude hold demonstration flight.

In addition to measuring distance from the ground, the IR sensors were used to determine position and heading of the vehicle relative to the environment. The filtered range estimates on the left and right facing IR sensors (see Figure 20) were used directly by the guidance algorithm for triggering different guidance states, as well as by the control algorithm for relative position control. The two forward-facing IR sensors were used together for estimating distance from a wall and heading relative to the same wall. As show in Figure 25, the left and right range readings on the forward sensors (denoted x_L and x_R) can be used to determine the vehicle relative heading, ψ, according to the equation:

$$\psi = \arctan\left(\frac{x_R - x_L}{L}\right) \tag{15}$$

where L is the distance between the two sensors. Even at moderate heading angles, the

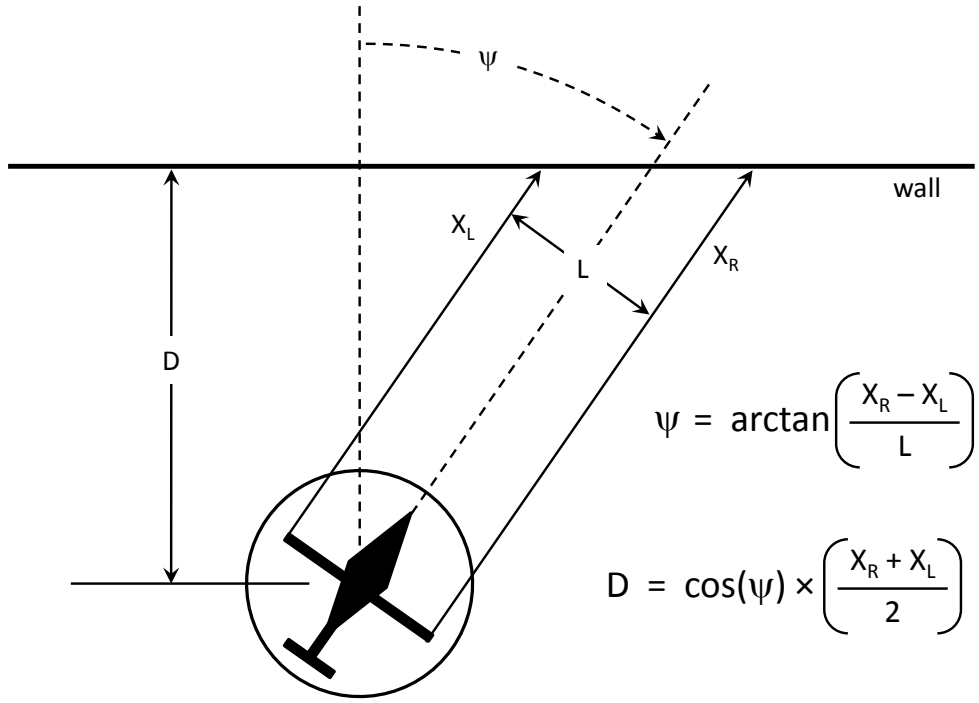


Figure 25: The two forward-looking IR sensors are used to calculate vehicle heading with respect to the wall (ψ). Relative heading and average range on the two sensors is used to calculate perpendicular distance to the wall (D). Note: the side-looking IR sensors are omitted for clarity.

range measurements can quickly approach the maximum useful range if the vehicle is flying a safe distance from the wall. Near maximum range, the IR sensors are characteristically very noisy, resulting in a noisy heading measurement. As the relationship between the range readings and ψ is nonlinear, the individual range readings were filtered as described above with estimates for x_L and x_R used to directly calculate ψ in leu of using an Extended Kalman filter to estimate the heading. The left and right range estimates were also used to estimate the distance to the wall, D, by using their average and the heading estimate as shown in Figure 25.

Figure 26 shows the performance of the wall distance and heading estimate using the two forward-looking IR sensors.

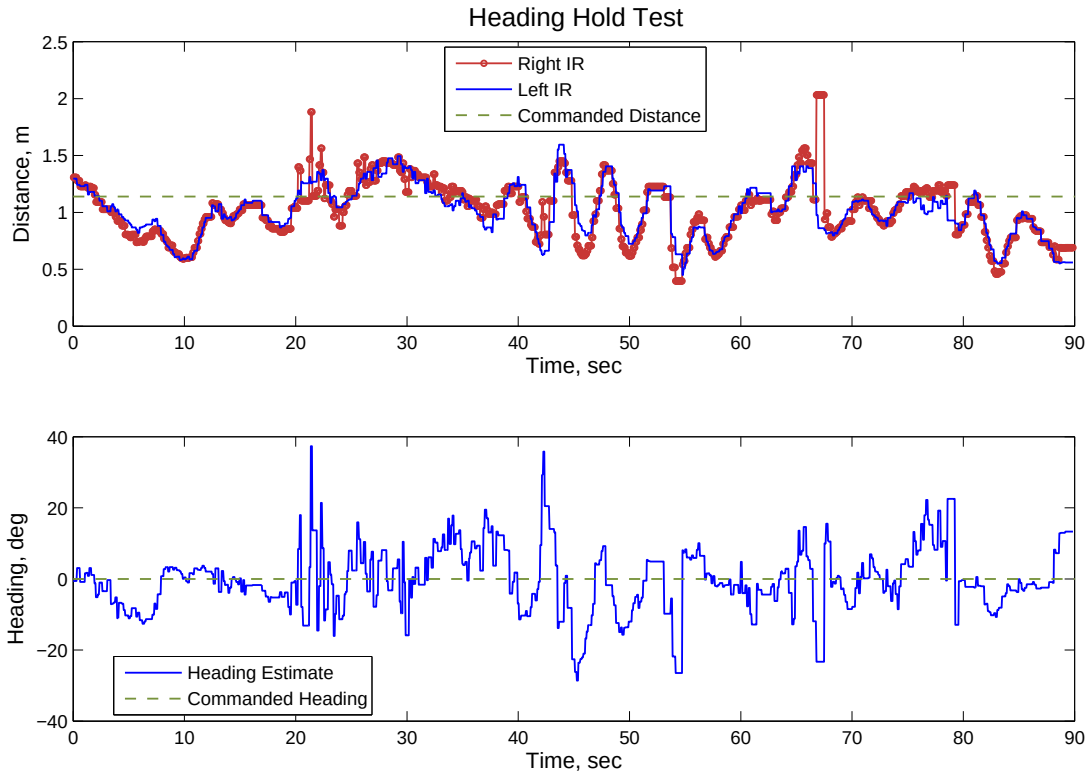


Figure 26: This test demonstrates the wall heading estimation and control algorithm, as well as the longitudinal distance control algorithm. During this test, the vehicle was flying laterally along a wall, at a commanded distance of 1.14m(45in), while trying to maintain a heading of zero degrees relative to the wall normal vector. The difference between the two forward looking IR sensors is used to estimate the heading, while the average of the two sensors is used to estimate the distance to the wall.

3.4 Control Algorithm

Hovering rotorcraft usually require a complex control system due to their inherent attitude instability. The system typical of helicopter UAVs [27] requires a complete state estimate including position, velocity, attitude, and angular rate in order to provide positive stability and position control. By using a vehicle with passive roll and pitch stability, measuring and controlling body rates is no longer required to achieve stable flight. Yaw stability is achieved by the use of a hobby heading-lock gyro, so no additional yaw rate stabilization is required by the controller. Thus, the traditional nested control loop architecture is not required, resulting in a vehicle control system that is greatly simplified.

The sensor combination described above does not take any inertial measurements, and

the vehicle guidance and navigation system do no require an estimate of the vehicle's inertial position or attitude. Since guidance and navigation occurs in the vehicle body frame, the control algorithm is simplified to four independent control loops [59]: altitude, longitudinal position, lateral position, and heading. The control architecture utilizes Proportional/Integral/Derivative (PID) design which assumes a linear vehicle dynamic model around a valid trim condition. The control loops seek to minimize the error between the commanded position (or reference model) and the measured position by generating appropriate servo commands. Range measurements are filtered as previously described to reduce noise, and a Kalman filter local velocity estimator is used in lieu of measured velocity for the derivative feedback. Integral feedback is provided by simply integrating the error over time.

A critical aspect affecting performance of the system is the determination of appropriate gains for each of the PID feedback loops. Gains for the altitude and heading loops were determined by trial and error during extensive flight testing. However, for lateral and longitudinal control, vehicle stabilization dynamics and aerodynamic interaction with the environment precluded a single choice for the control gains. During flight testing, the vehicle was observed to behave differently depending on the distance to nearby walls. At distances approaching 1.0m from a wall, the vehicle had a tendency to push away the wall. At distances closer than approximately 0.5m, the vehicle showed a tendency to be drawn toward the wall, usually resulting in contact. As a result of this behavior, a method of gain scheduling was developed whereby different gains were used depending on the vehicle's proximity to obstacles [59]. Variable aerodynamic effects were also encountered during ground effect flight, however gain scheduling was not required in the altitude loop since commanded autonomous hover altitudes were above the maximum ground effect height (approximately 0.5m for this vehicle).

The feedback control algorithm is summarized below for each of the four independent control actions:

1. **Altitude Hold:** The altitude loop uses the filtered range measurements from the downward pointed sonar for altitude control. A PID architecture is used, where the derivative of the position is calculated using a Kalman filter as described previously.

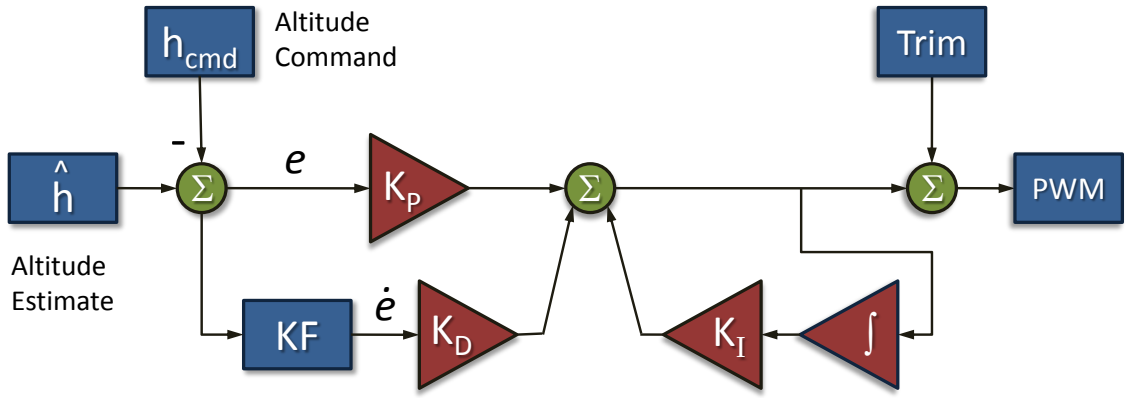


Figure 27: PID architecture for the altitude hold controller. Longitudinal and Lateral control loops use a similar architecture.

During vehicle operation, varying battery voltage level affects the throttle trim value. The integral part of the controller is used to counteract this effect. Rather than integrating the position as is traditionally done, the servo commands output by the controller are integrated instead. As a result, the system can inherently handle actuator saturation and integration windup. Furthermore, servo commands are easier to measure since they are assigned by the controller. Figure 27 shows the schematic of the altitude control loop. The lateral and the longitudinal control loops have a similar architecture.

2. **Heading Hold:** The vehicle's heading relative to the facing wall is measured as described in the navigation section above. The filtered range measurements from two forward-facing IR sensors are used to calculate heading according to Equation 15. When the guidance system is in inside or outside turning mode, the vehicle is placed in an open-loop turn until the range sensors detect completion of the turn. When the vehicle returns to wall-following mode, the heading hold controller is re-engaged. The IR range measurements are very noisy near maximum range, so vehicle's distance from the facing wall is controlled to keep the vehicle from flying too far from the wall.

3. **Longitudinal Position Control:** The longitudinal position control loop is used to

ensure that the vehicle maintains a fixed distance from a wall or obstacle in the longitudinal direction. Control is achieved by using the average of the two filtered range readings from the forward-facing IR sensors. In order to improve the longitudinal position and heading measurements, the distance from the facing wall is kept in the range of 0.5m-1.5m. Above this distance, the IR sensors are unreliable, so the vehicle is commanded to perform slow open-loop forward flight until the wall is within range. If the vehicle is below minimum range and moving toward the wall, gains are increased to prevent aerodynamic effects from causing the vehicle to be drawn against the wall. Between minimum and maximum range, the gain is interpolated for smoother performance at intermediate ranges. Other than this gain scheduling, the architecture of the longitudinal position control is similar to that of the altitude hold controller.

4. **Lateral Position Control:** The lateral position control loop is used to detect and avoid obstacles in the lateral path of the vehicle during wall-following flight. When a wall or obstacle is detected as the vehicle flies along the wall, the filtered range measurements from the IR sensor facing in the direction of flight are used to slow the vehicle and prevent it from hitting the wall. in addition, the guidance mode is switched to "Inside Turn" mode when the desired minimum lateral range from the wall is reached. The control loop architecture is similar to the altitude hold controller.

3.5 Simulation Results

Two different simulations were developed during the design and testing of the GNC algorithms for the coaxial helicopter vehicle. First, the vehicles and sensors were modeled using the open-source software Blender [4]. The purpose of this simulation was to determine the best combination and placement of range sensors, as well as to develop the guidance algorithm logic. First, the vehicle and environment were modeled graphically. Next, a vehicle dynamics model was created and refined based on experience from manual flights of the coaxial test vehicle. Finally, IR and sonar range sensors were added to the vehicle model, and initial guidance and navigation algorithms were developed. The Blender simulation allowed for easy configuration of different test environments, with variable room size and

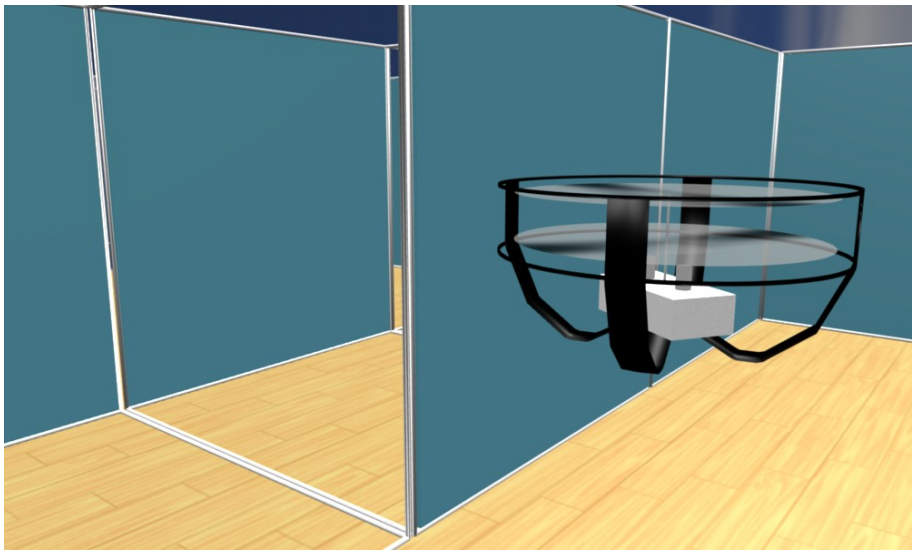


Figure 28: Screen capture from Blender simulation.

obstacle placement. Variations on the guidance algorithm logic were easily tested before attempting to fly the actual hardware. In addition, some mapping and localization algorithms were developed using the simulation as described in Appendix A. Figure 28 shows a screen capture from the Blender simulation.

Once the vehicle was built and flight testing had begun, a higher fidelity simulation was developed using software developed in-house at the Georgia Tech UAV Research Facility. The primary purpose of this second simulation was to test the actual flight code in simulation before flying it on the vehicle. This simulation had a more accurate vehicle dynamic model, sensor performance that was based on the characterization experiments discussed above, as well as a simulated test environment. Since the GNC software that was developed for the flight computer was the same code used to control the vehicle in simulation, the guidance, navigation, and control algorithms, as well as control system gains, were further refined during the flight test program through simulation prior to actual flight. Figures 29 and 30 show the flight vehicle simulation during successful autonomous navigation of a simulated building.

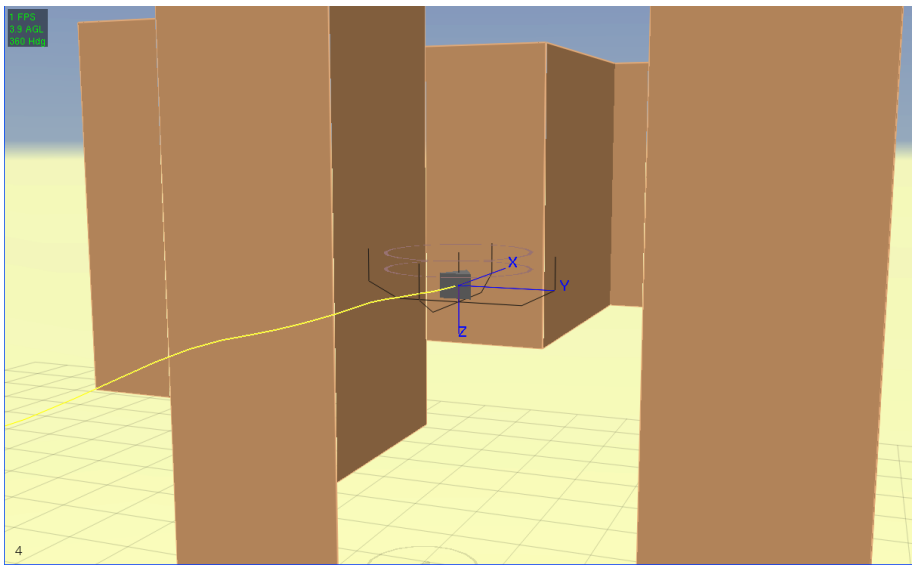


Figure 29: Screen capture from flight vehicle simulation.

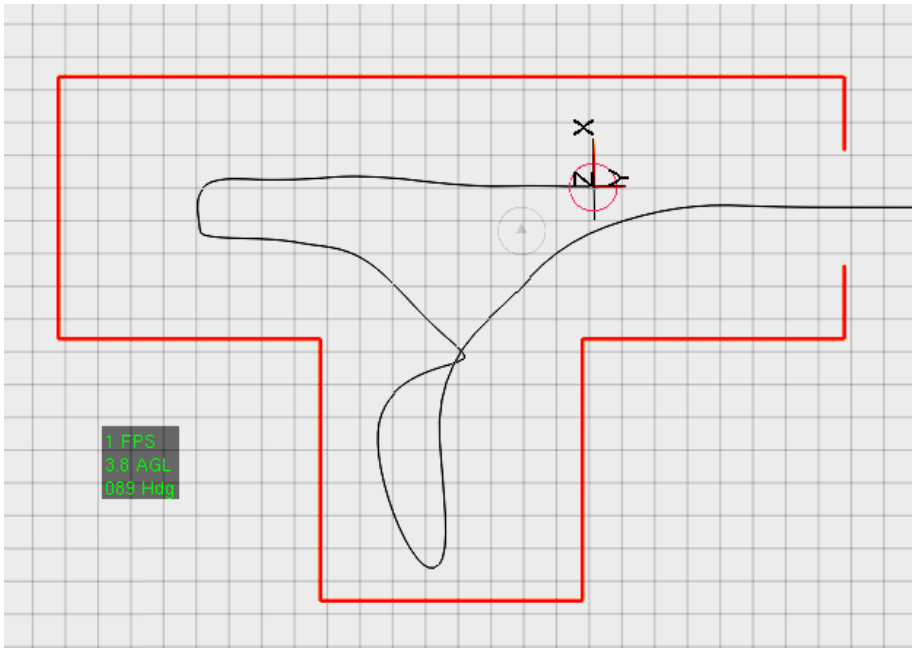


Figure 30: Screen capture from flight vehicle simulation showing navigation solution. The vehicle path is traced during simulation to show where the vehicle has flown during the mission.

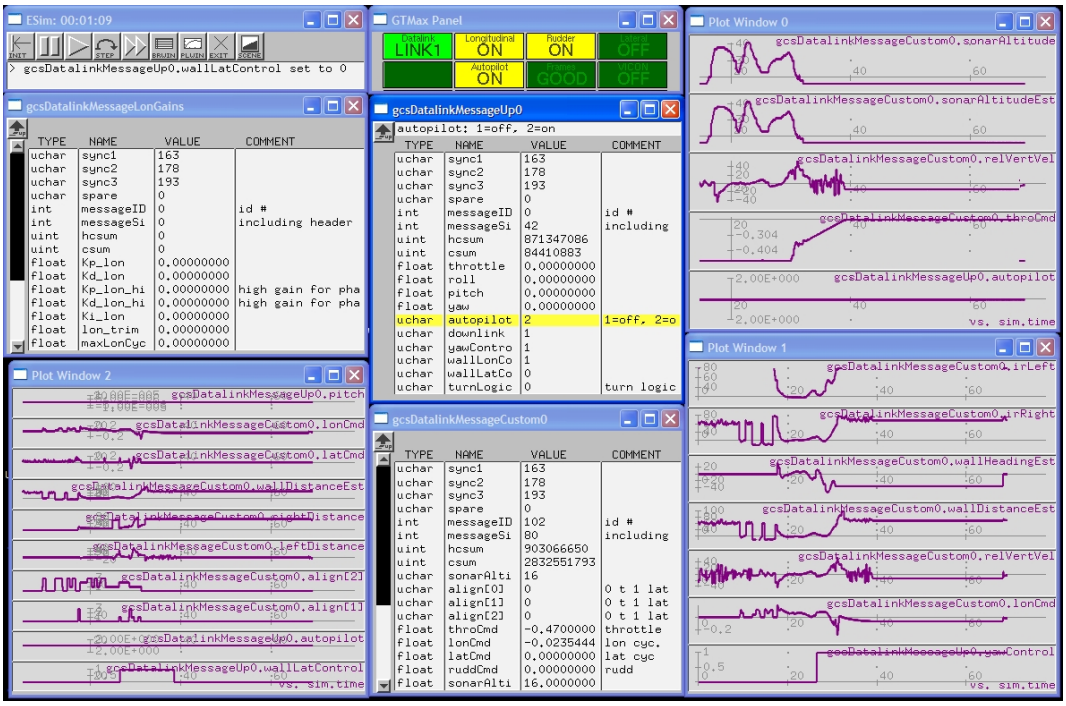


Figure 31: Screen shot of ground control station.

3.6 Flight Test Results

During flight testing, all navigation, guidance, and control algorithms were performed onboard the ATMega128 microprocessor. Thus, the controllers operated at a the same rate as the sensor measurements, while the range data and vehicle status was transmitted to the ground at a rate of 10Hz for monitoring and post-flight analysis. A Ground Control Station (GCS) was developed to enable real-time monitoring of all sensor data, adjustment of control loop gains during flight, and as a pilot interface during manual flight. In addition to real-time monitoring, all sensor data and vehicle status information is recorded during flight for later analysis. A screenshot of the GCS is shown in Figure 31. A separate computer was also used to view and process imagery collected via an onboard camera, which was required to complete the IARC mission. Although it is possible to use the camera for measuring velocity (via optical flow) or possibly position [69], such methods were not employed by this system.

A typical flight test began with manual takeoff and flight to a desired altitude. Next, the altitude control loop was enabled, bringing the vehicle to a commanded altitude while

all lateral and longitudinal controls were still managed by a remote pilot. The altitude loop gains were tuned very early in the flight test program, and its performance was superb, so altitude control was generally enabled during independent testing of all the other control loops. After altitude control was proven, control loops were closed incrementally, with PID control gains and gain schedules tuned as each new capability was added. To establish a rough order of magnitude on the gains, the heading, longitudinal, and lateral control loops were each closed individually. Then, the loops were added one at a time to further tune the gains. First, yaw control was enabled so that the vehicle could maintain a fixed heading with respect to a facing wall while the vehicle was under manual control in the lateral and longitudinal directions. Once the heading control loop was working well, the longitudinal loop with gain scheduling was tuned. Finally, the lateral control loop was tuned such that the vehicle could then hover in a room at a fixed position relative to a facing wall without drifting into any walls in the lateral direction.

Once the control loops were closed, the guidance algorithms were enabled and the vehicle trim conditions were tuned such that the vehicle could maintain a desired range from a wall while flying laterally along the wall. The guidance logic was adjusted to enable the vehicle to make inside and outside turns within the building structure while exploring its environment as described above. Flight times lasted approximately eight minutes, with all range sensors, the microcontroller, 60mW 2.4GHz datalink, and a camera with dedicated 1W video link operating simultaneously.

During the ongoing flight test program, the vehicle was flown in the 2009 International Aerial Robotics Competition. The competition allowed four attempts to navigate a maze constructed in a basketball arena. Each attempt was initiated by enabling completely autonomous control from a distance of 3m outside the competition arena. This distance was outside the range of the IR sensors used for heading and longitudinal control, so the vehicle flew forward at a predetermined speed until the arena structure was detected. The forward-looking pair of IR sensors were used to adjust lateral position of the vehicle with respect to the 1m square entry window, while the side-looking IR sensors were used to determine when the vehicle had entered the arena. Once the vehicle was inside, the wall-following guidance

mode was triggered. The vehicle performed two successful autonomous flights through the window into the test arena, followed by autonomous wall-following behavior within the test arena. During a third flight, the vehicle missed the target window and began executing its wall-following guidance algorithm on the outside of the arena. Successful altitude hold, wall-following behavior, and inside and outside turns were all demonstrated during fully autonomous flight at the 2009 IARC, resulting in a second-place award in the competition overall. After the competition, the flight test program resumed at Georgia Tech in order to collect more data and further refine the GNC algorithms.

An indoor test environment was designed to further tune the guidance turning modes and the transition between turning modes and the wall-following mode. The wall arrangement, shown in Figure 32, included a series of inside and outside turns, separated by a straight length of wall that was long enough to demonstrate the lateral wall-following mode with heading control. An example of longitudinal distance control and heading control is shown in Figure 26. The altitude control performance is shown in Figure 33, while the longitudinal and lateral control for a typical flight test is shown in Figure 34. Approximately 10 fully autonomous flights were performed in the test environment, with several dozen flights in all used to tune the various GNC algorithms.

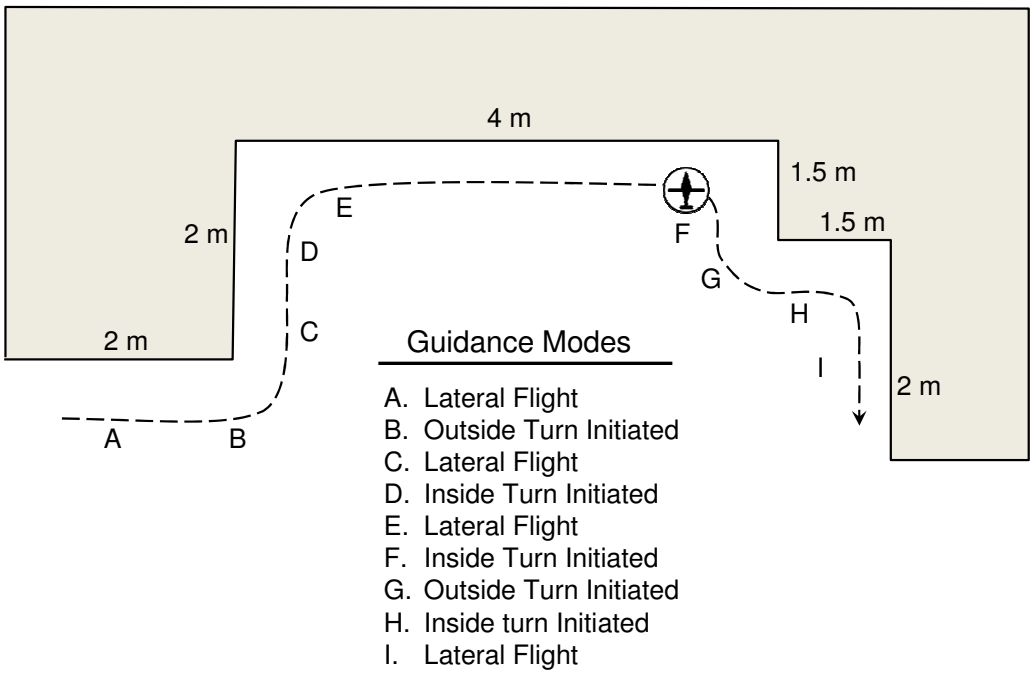


Figure 32: Typical flight test environment. In this test, the vehicle begins at the left side of the room and flies to the right while maintaining a specified distance from the wall. When outside and inside turns are encountered, the vehicle guidance system switches state and performs the appropriate behavior.

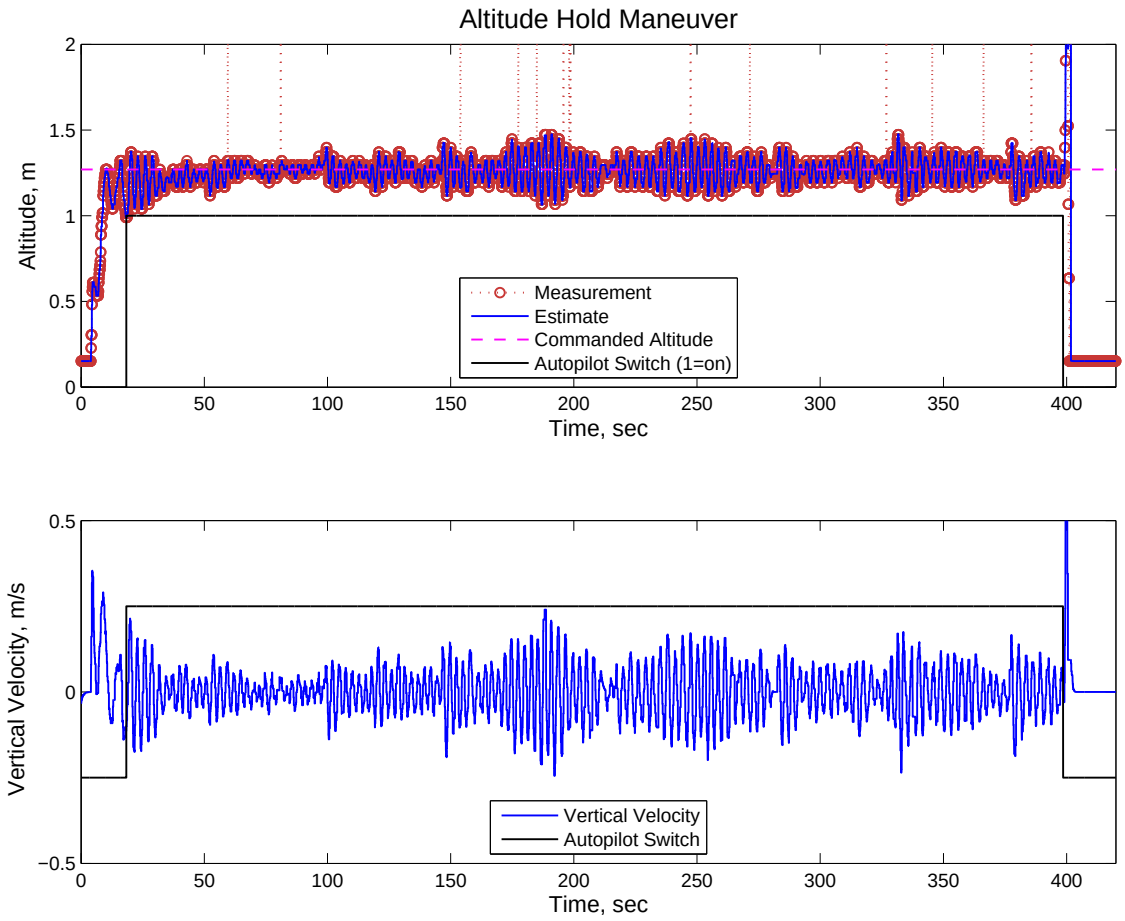


Figure 33: Altitude measurements and estimate, with vertical velocity estimate. Commanded altitude was 1.14m (45in). The autopilot command represents the enabling and disabling of the altitude-hold controller.

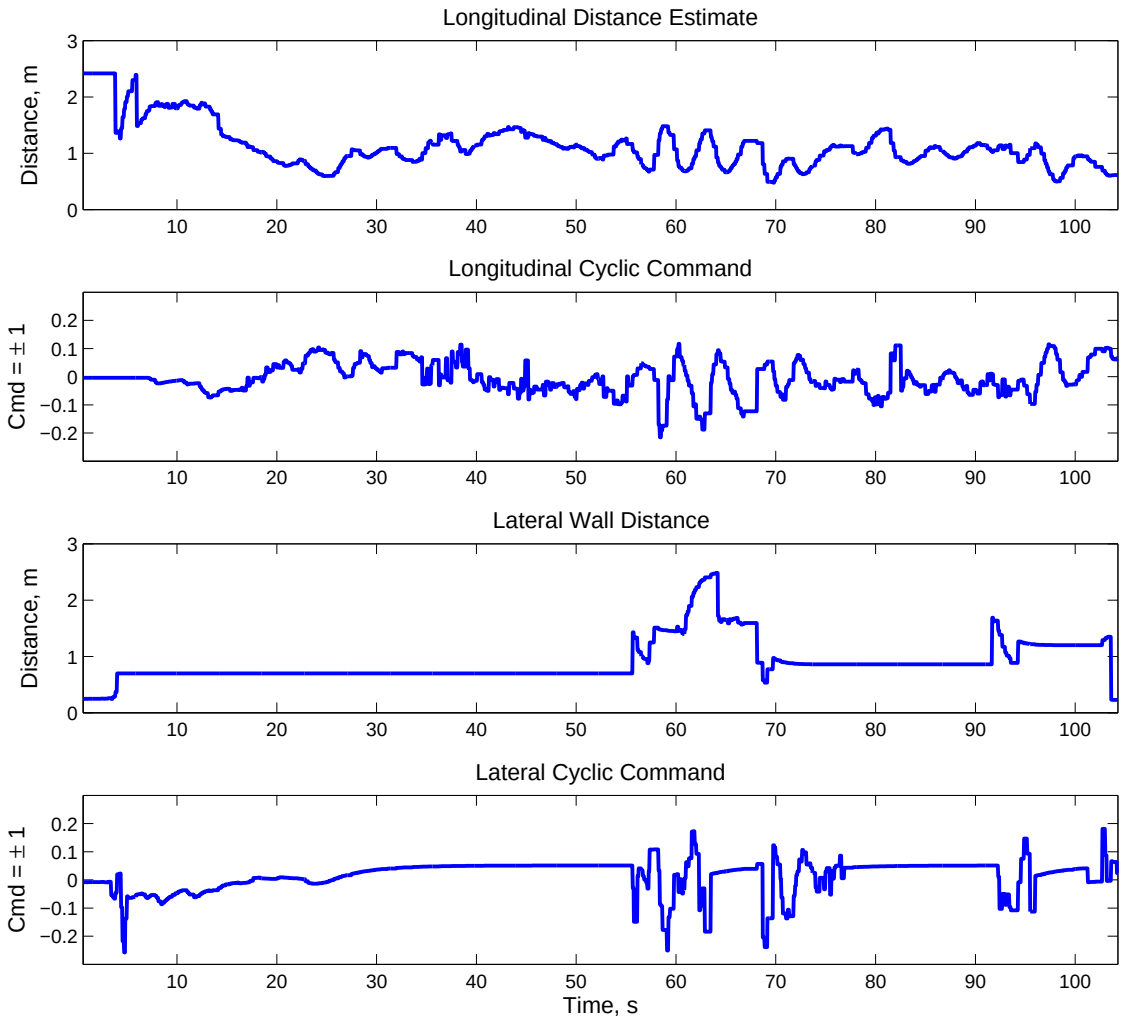


Figure 34: Longitudinal and lateral distance and cyclic commands recorded during a typical flight test.

CHAPTER IV

LASER AIDED INERTIAL NAVIGATION

Missions that require UAVs to enter unknown environments, including structures, may not have reliable reception of radio signals. As a result, reliance on GPS for navigation, or on ground computers performing complex GNC algorithms with commands relayed to the vehicle, limits the environments that can be effectively explored. This chapter describes a navigation system that utilizes a scanning laser range sensor and sonar altimeter in lieu of GPS for position updates to a traditional EKF-based state estimation algorithm. A description of the vehicles and sensors compatible with this system is provided, followed by a discussion of two different candidate algorithms for analyzing laser scan data to estimate vehicle motion. A novel method is presented for determining the uncertainty of pose estimates related to the topology of the environment, as well as recursive method for extracting line segments from scan data. Guidance, navigation, and control algorithms are implemented in simulation, as well as experimentally, with results provided at the end of the chapter.

4.1 Vehicles and Sensors

4.1.1 Aerial Vehicle

Simple indoor navigation has been demonstrated using a passively stable coaxial helicopter, but maneuverability is limited and the vehicle dynamic response is much more complicated [59]. Any autonomous indoor flight vehicle must also be capable of carrying the required sensors and onboard computer required for navigation. Figure 35 shows some potential vehicle configurations compatible with autonomous indoor flight using laser-augmented inertial navigation.

(a) Helicopter Concept (b) Quadrotor Concept

Figure 35: Conceptual vehicles designed for laser aided inertial navigation. Vehicle major dimension is approximately 80 cm.

4.1.2 Sensors

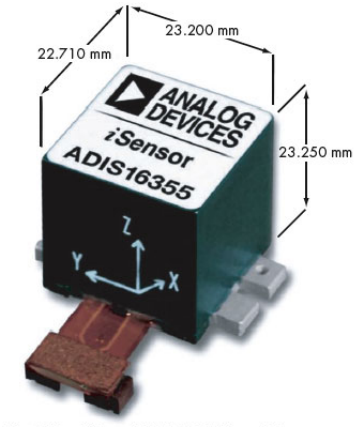
The primary necessity for indoor navigation using an unstable vehicle is to provide a position reference to bound the drift in the navigation solution. As previously discussed, vehicle motion can be estimated by using an Extended Kalman Filter with sufficiently accurate IMU data and a dynamic model of the vehicle. Drift in this estimate requires correction by an additional position measurement. This research addresses one approach to measuring vehicle position directly through the use of laser scan data for taking range measurements in the plane of the vehicle's flight. The scan data is compared to a map of the environment which has been created using past scan data. The process has been demonstrated in simulation to sufficiently replace GPS for indoor navigation of a helicopter equipped with an IMU, a sonar for altitude measurement, and a scanning laser rangefinder. The scanner itself consists of an infrared laser which is reflected off a spinning mirror, causing the beam to pan across the environment. The reflected laser light returns to the sensor and is detected by the sensor, which uses the reflection to triangulate the range. The return is sampled at fixed intervals, resulting in a series of range measurements with regular angular separation. A common practice is to consider the laser scan as a set of 1-dimensional range readings in hundreds of directions that span the sensor field of view. As a practical matter, this assumption treats the angle of measurement as an independent variable whose uncertainty is neglected, while the range reading is a dependent variable with a known uncertainty in the measurement direction [2, 43]. Implications using this essentially 1-dimensional error

to calculate the uncertainty of a 2-D position and heading estimate are discussed below. Figure 36 shows some examples of small, lightweight, commercially available sensors that could be used for indoor navigation via the methods described below. These sensors were modeled in simulation and used in experimental testing as described in the sections below.

4.2 Laser Aided Navigation Algorithms

The laser scanner model and simulation environment can be used to test a desired SLAM algorithm in lieu of using actual hardware. A variety of SLAM algorithm implementations are available for free use at the web site OpenSLAM.org. The algorithm used for the preliminary research, called CoreSLAM [62], was chosen primarily because it is simple, easy to implement, and it uses integer math where possible to improve computational speed [61]. There are two main parts to any SLAM routine. The first task is to measure distance to obstacles or landmarks in the environment, and to map them given the vehicle's position and orientation (i.e. mapping). The second task is to determine the best estimate of the vehicle's position and orientation based on the latest scan (or series of scans) given a stored map (i.e. localization). The mapping and localization tasks are performed together to maintain the most current map and position estimate. In this research, two different algorithms for localization and mapping were tested in simulation as well as experimentally. First, a simple open-source algorithm called CoreSLAM [62] was tested, and later an Iterative Closest Point (ICP) scan-matching algorithm was developed in-house. As discussed in Chapter 2, in the robotics and computer science community much emphasis is placed on keeping track of vehicle motion and solving chains of pose constraints between differen locations to make corrections to a global map. For basic indoor navigation, however, it was determined that this level of accuracy in a global map is not required, or even desired if it comes at too steep a computational burden. As a result, the algorithms discussed here are primarily designed for localization with respect to the immediate environment, not to building and maintaining highly accurate global maps. As such, the CoreSLAM mapping routine as implemented here does not detect or correct errors in past observations, and the "map" in the ICP algorithm consists entirely of a singe previous scan. However, with the ICP algorithm individual scans

(a) MaxBotix® EZ1 Sonar



(b) Analog Devices ADIS 16355 IMU

(c) Hokuyo URG-04LX-UG01 laser scanner

Figure 36: There are many lightweight commercially available sensors which would be useful for indoor navigation on UAVs.

can be saved and post-processed into a global map if desired.

4.2.1 CoreSLAM

In CoreSLAM, estimated vehicle state information is used to align each new scan with the evolving map. CoreSLAM maintains a map that consists of a two-dimensional array containing integer values ranging from 0 to 65535. The map is initialized to a middle value of 32768, and values are adjusted as each new scan is processed to reflect the evolving map. In order to easily visualize the map, the values are scaled to the range 0-255 and displayed on a one-to-one scale 8-bit gray scale image. As observations are made, areas where obstacles are detected are darkened, and areas that are clear of obstacles increase in brightness. The map image thus represents an occupancy grid, with color value displaying the probability that a square is occupied. As more areas are explored, the observed obstacles are shown by darkened areas on the map, while clear areas are displayed by lighter colored pixels.

For the experiments described below, a map size of 400 with a scale of 0.2 feet was used. Thus, the displayed map is 400 x 400 pixels, displaying 80ft^2. When the mapping algorithm is running, the map is updated at a user-defined rate (currently 0.1 Hz) using the current position and orientation estimate as described in [61]. Figure 38(b) shows an example of the map produced by the CoreSLAM routine during simulation.

Once the map is initialized, the algorithm continuously monitors and incorporates new scan data into the map. The scan data is transmitted from the sensor as a vector of range measurements, where the angle increments counterclockwise. CoreSLAM requires the scan data to be in a Cartesian coordinate system, so the scan data is first converted from polar coordinates. Next, the position estimate is updated if desired. The vehicle always maintains an estimate of its position in the navigation state vector, and this estimate is used as a starting point for the localization routine. A scan registration algorithm is used to determine the most likely vehicle position and orientation (pose) for a given scan. This is accomplished by performing a Monte Carlo search of different poses nearest the current pose estimate, and evaluating the latest scan at the new pose to see how closely it matches the most current map. The current pose estimate is updated by using the vehicle's EKF

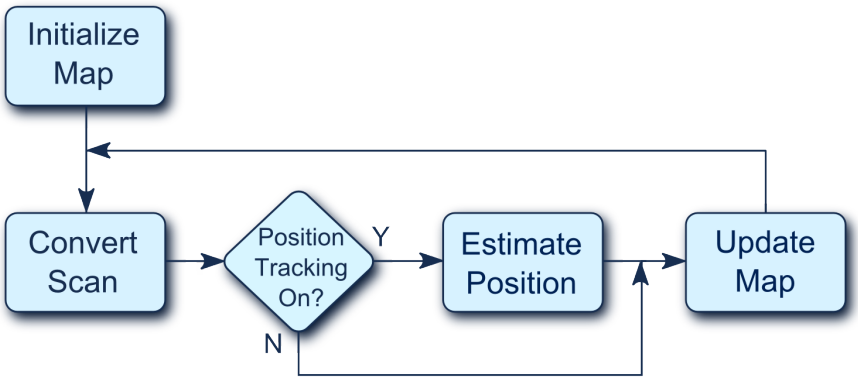
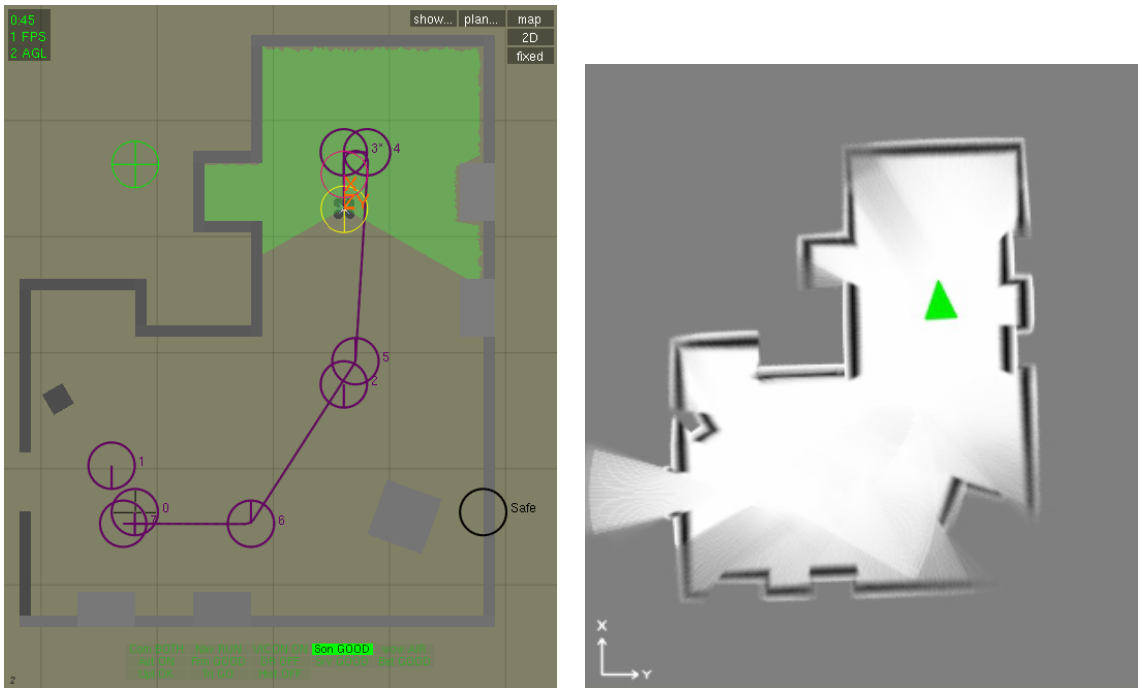


Figure 37: This flowchart shows the CoreSLAM algorithm as implemented in simulation.

navigation solution to provide a more accurate starting point for the Monte Carlo search. If the scan matching routine can find a better pose to match the current scan with the most recent map, the current pose is updated to the new pose. In [61], the measure of how well a particular scan matches the latest map is described as the "distance" between the scan and the map. The Monte Carlo routine calculates the distance between the scan and the map and returns the pose that minimizes this distance. Next, the scan data is incorporated into the map using the updated pose. Figure 37 shows a flow chart that describes how the CoreSLAM mapping and localization algorithms are implemented in the simulation.

Figure 38(a) shows a simulated building interior being explored by a helicopter with a scanning laser rangefinder. Figure 38(b) shows the map generated during a simulated flight.

For this research, the CoreSLAM algorithm was modified and improved in two important ways. First, the position covariance matrix of the navigation solution was used as an input to the map update function. In the original algorithm, a user-defined constant value was used to create the Gaussian uncertainty on obstacle locations. To prevent unrealistic confidence in the map, the actual sensor range uncertainty (which is a function of range detected) was added to the vehicle's position uncertainty. The map update algorithm was then modified to use this total uncertainty when assimilating scan data into the map. A second improvement made to the CoreSLAM algorithm was to use the vehicle's state estimate and covariance as inputs to the Monte Carlo search. Thus, the random search initial conditions and search scope were significantly improved over the original algorithm by providing a good starting



(a) Mapping a simulated environment

(b) Map generated during simulation

Figure 38: The map maintained by the CoreSLAM routine represents an occupancy grid, where the value of each pixel in the image represents the likelihood that a particular grid square is occupied. Here, lighter colors represent free space, while darker colors represent obstacles and medium gray areas are unexplored. Areas with higher contrast represent greater certainty due to longer observation periods during the flight. The green triangle represents the vehicle's estimated position and heading.

point for the search, resulting in better match performance.

4.2.2 Iterative Closest Point Scan Matching

In contrast to the probability map used by CoreSLAM, the Iterative Closest Point (ICP) algorithm does not use a complete global map. Strictly speaking CoreSLAM uses scan registration (a scan compared to a map) to estimate vehicle pose relative to the map, while ICP uses scan matching (a scan compared to another scan) to estimate vehicle pose change relative to a previous scan. With regard to SLAM, in this custom ICP algorithm the "map" consists entirely of a single scan collected and stored at a previous time. As the vehicle moves and collects new scans of the environment, the "map", or key frame scan, is replaced by a more recent scan. Updates of the key frame scan occur based on user-adjustable parameters that include distance the vehicle has traveled and a measure of how well current scans correspond to the key frame scan.

The ICP algorithm begins by making a copy of the key frame scan and the current scan that are rotated and translated to the inertial frame using the pose estimate (x, y, θ) for each scan. Then, a nearest-neighbor match is performed to determine which points in the current scan correspond to points in the key frame scan. Next, the pose error $(\Delta x, \Delta y, \Delta\theta)$ between the current scan and the key frame scan is determined. The pose difference is calculated to minimize the least squared error between the set of corresponding points in the current and key frame scans. Equation 16 shows the rotation matrix relating the i^{th} point of the current scan to the corresponding i^{th} point of the key frame scan.

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix}_k = \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) \\ \sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}_c \tag{16}$$

Here, the subscript k indicates the key frame scan points expressed in the inertial frame, c represents the current scan points expressed in the inertial frame, and $\Delta\theta$ represents the rotation difference between the current and key frame scans. With some re-arrangement of terms, solving for the rotation angle can be set up as a constrained linear least-squares

problem, where the nonlinear constraint equation is: $\sin^2(\Delta\theta)+\cos^2(\Delta\theta)=1$. Equation 17 shows the more practical arrangement, where $\cos(\Delta\theta)$ and $\sin(\Delta\theta)$ are parameters to be estimated, and the $(x_i, y_i)_c$ scan points are measurements of these parameters.

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix}_k = \begin{bmatrix} x_i & -y_i \\ y_i & x_i \end{bmatrix}_c \begin{bmatrix} \cos(\Delta\theta) \\ \sin(\Delta\theta) \end{bmatrix} = \mathbf{H} \begin{bmatrix} \cos(\Delta\theta) \\ \sin(\Delta\theta) \end{bmatrix} \tag{17}$$

This relationship is identical for each scan point, such that the scan coordinates may be stacked up into a vector as shown in Equation 18.

$$\begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{bmatrix}_k = \begin{bmatrix} x_1 & -y_1 \\ y_1 & x_1 \\ \vdots & \vdots \\ x_n & -y_n \\ y_n & x_n \end{bmatrix}_c \begin{bmatrix} \cos(\Delta\theta) \\ \sin(\Delta\theta) \end{bmatrix} = \mathbf{H} \begin{bmatrix} \cos(\Delta\theta) \\ \sin(\Delta\theta) \end{bmatrix} \tag{18}$$

The solution that minimizes the squared error is shown in Equation 19. Fortunately, due to the structure of the $\mathbf{H}$ matrix, this equation can be reduced analytically to the form shown in Equations 20 and 21. A similar result was presented by Martínez in [40], although it is derived there by defining a cost function and minimizing with respect to squared error. Martínez also provides an excellent overview of additional scan matching techniques that have been used successfully in the past, although few can match ICP for its simplicity and effectiveness for finding local minimum matches between laser scans [40].

$$\begin{bmatrix} \cos(\Delta\theta) \\ \sin(\Delta\theta) \end{bmatrix} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T \begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{bmatrix}_k \tag{19}$$

$$\begin{bmatrix} \cos(\Delta\theta) \\ \sin(\Delta\theta) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} x_{c_i}^2 + y_{c_i}^2 & 0 \\ 0 & \sum_{i=1}^{n} x_{c_i}^2 + y_{c_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} x_{c_i} x_{k_i} + y_{c_i} y_{k_i} \\ \sum_{i=1}^{n} x_{c_i} y_{k_i} - y_{c_i} x_{k_i} \end{bmatrix} \tag{20}$$

$$\cos(\Delta\theta) = \frac{\sum_{i=1}^{n} x_{c_i} x_{k_i} + y_{c_i} y_{k_i}}{\sum_{i=1}^{n} x_{c_i}^2 + y_{c_i}^2} \qquad \sin(\Delta\theta) = \frac{\sum_{i=1}^{n} x_{c_i} y_{k_i} - y_{c_i} x_{k_i}}{\sum_{i=1}^{n} x_{c_i}^2 + y_{c_i}^2} \tag{21}$$

Given the form derived here, the angle that minimizes the squared-error of rotation difference between the current scan and the key frame scan can be found by simply summing the product of scan point coordinates without resorting to inverting or multiplying matrices. This results in a very efficient estimation algorithm. In fact, the denominator in Equation 21 need not be calculated to find $\Delta\theta$ since it divides out during the arctangent function shown in Equation 22. Once $\Delta\theta$ is estimated, Δx and Δy can be found by averaging the error between the key frame scan points and the corresponding current frame scan points (corrected by a rotation of $\Delta\theta$). The solutions for Δx and Δy are found below in Equation 23.

$$\Delta\theta = \arctan\left(\frac{\sum_{i=1}^{n} x_{c_i} y_{k_i} - y_{c_i} x_{k_i}}{\sum_{i=1}^{n} x_{c_i} x_{k_i} + y_{c_i} y_{k_i}} \right) \tag{22}$$

$$\left.\begin{aligned} \Delta x &= \left(\tfrac{1}{n}\right) \sum_{i=1}^{n} x_{k_i} - \left(x_{c_i} \cos\Delta\theta - y_{c_i} \sin\Delta\theta\right) \\ \Delta y &= \left(\tfrac{1}{n}\right) \sum_{i=1}^{n} y_{k_i} - \left(x_{c_i} \sin\Delta\theta + y_{c_i} \cos\Delta\theta\right) \end{aligned}\right\} \tag{23}$$

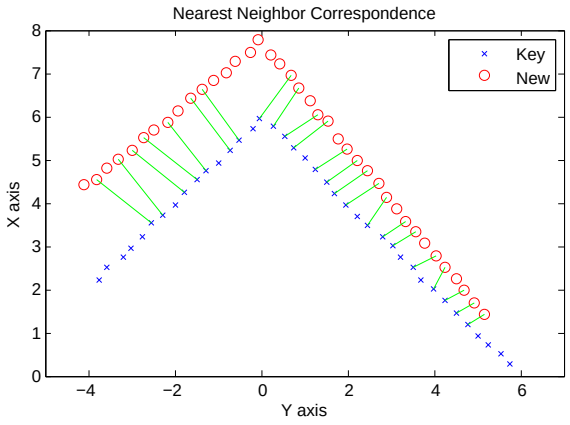
One benefit of this simplified form is that the ICP scan matching routine is very fast. The number of calculations required for the closest point scan match is on the order of N, the number of corresponding points between the two scans. In fact, as Martínez noted [40],

the most computationally expensive part of the ICP algorithm is the nearest neighbor match, which requires on the order of N^2 operations. The entire process of determining scan point correspondence and finding the pose error $(\Delta x, \Delta y, \Delta\theta)$ is iterated until a user-defined minimum root-mean-squared error between two sets of corresponding scan points is achieved. Although this error is a useful metric for determining how well the scan matching algorithm was able to align two scans, it is not a good metric for the uncertainty of the pose estimation that results. A better method for measuring pose estimate uncertainty, which includes an analysis of the local topology, is provided in sections below. Figure 39 shows an example of the ICP scan matching process.

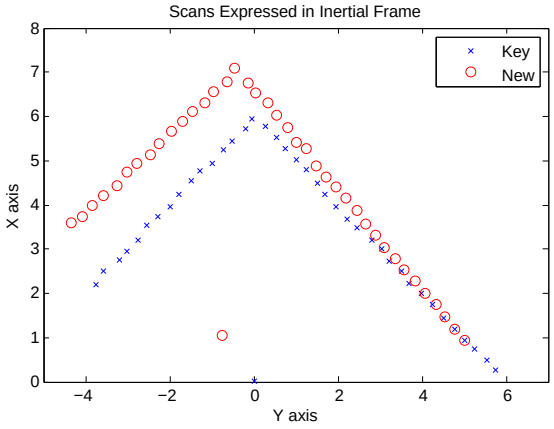
4.3 Laser Scan Information Content

Scanning laser range finders produce a rich data set that can be exploited in ways other than vehicle odometry based on scan-matching or scan-registration. The actual *shape* of the surrounding environment can be deduced with the proper analysis, and features can be extracted from the data to improve the vehicle's performance. Feature point detection and tracking is a common machine vision task, although image processing can be somewhat computationally expensive. For laser scan data, however, features such as straight walls, corners, and simple curves can be easily and quickly identified [67]. In this research features were restricted to line segments, limiting the possible algorithms to the fastest ones that still produced the necessary information to accomplish all guidance and navigation tasks. The line segment estimates were used in two different ways, both contributing to the success of the navigation system. First, each incoming scan was modeled as a series of line segments to determine the orientation of continuous walls. The guidance algorithm can use this information to align the vehicle with any feature in its immediate surrounding and thus fly at any desired orientation with respect to the observed walls. Any gaps in the walls are identified, and the guidance system can then determine whether or not to attempt to fly through gaps that are deemed large enough (i.e. doors or hallways) or simply ignore them.

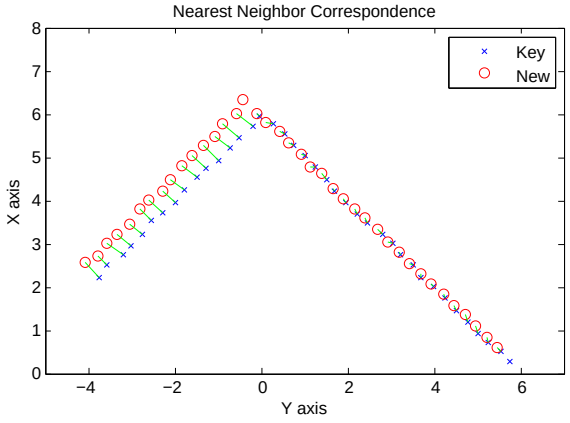
The second feature analysis technique developed here pertains to the uncertainty of the pose estimates produced by the ICP or CoreSLAM algorithms. Any scan-matching



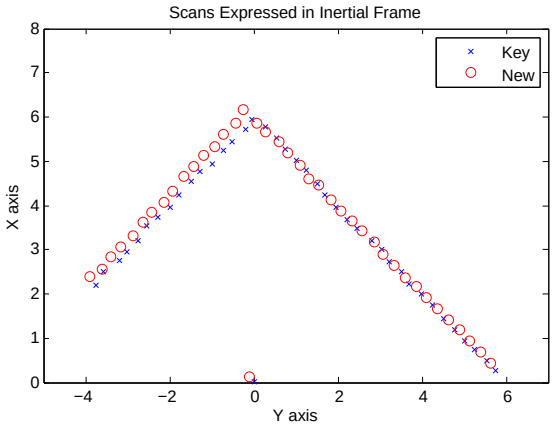
(a) First correspondence



(b) First correction



(c) Fifth correspondence



(d) Fifth correction

Figure 39: The ICP scan matching process. First corresponding points are paired, then the transformation (Δx,Δy,$\Delta\theta$) that minimizes the squared error is determined and applied. The process iterates until no better match is found (in this example, after five iterations). Note that in this case, a local minimum has been found due to improper correspondence of the points along the top right of the scan.

algorithm can provide the "fit quality" as well as the pose estimate. For ICP, this can be deduced from the squared-error summed over the corresponding scan points. CoreSLAM uses the sum of the map probabilities for each scan point as a measure of the scan fit. However, these and all similar algorithms fail to identify the uncertainty due to the *structure* of the environment. For example, a vehicle in a long, straight hallway will may have a good estimate of its position with respect to the walls on each side, but very little certainty of its position along the length of the hallway. Similarly, a vehicle in the middle of a circular room might have a good position estimate, but estimating heading would be impossible. In these situations, a scan-matching algorithm may produce a fit that has very little error, while at the same time providing very little information in certain directions. This Dilution of Precision (DOP) problem causes a position estimate that relies solely on the scan-match fit quality to quickly become overconfident in directions where few feature are observed. To solve this problem, a novel method was developed to identify the uncertainty in the x, y, and θ directions based on the orientation and distance to line segments extracted from the matched scan points (or in the case of CoreSLAM, the entire scan).

4.3.1 Line Extraction

A nearly exhaustive survey of line extraction techniques was performed by Nguyen et al [44] which identified a few algorithms that excelled in speed, robustness to outliers, and ability to correctly identify lines in a set of experimentally collected laser scan data. Analysis and experimental results showed that two methods outperformed the others: a recursive algorithm commonly known as *Split and Merge (SM)* or *Iterative End Point Fit (IEPF)* and an incremental approach commonly referred to as *Line Tracking (LT)*. As a result, these two methods were analyzed for suitability with respect to the aerial indoor navigation problem.

Two line extraction algorithms were implemented and tested in MATLAB® using scan data collected during this research. The SM algorithm was implemented essentially as described in [44]. The process begins by connecting the end points of the sequentially collected scan data to form a line segment. Next, the perpendicular distance of each point

from the original segment is calculated, and the point that is farthest from the segment is calculated. If this distance is above some threshold, the original line segment is split at the point of maximum distance, forming two new line segments. In this implementation, the sensor standard deviation on range was used to determine the split threshold. The process is repeated recursively, with each new segment being split into two new segments until all of the data is assigned a segment meeting the threshold requirement, with segments shorter than a specified length being eliminated. The algorithm for creating line estimate parameters (described in detail below) has complexity on the order of N, where N is the number of scan points to be fit to the line, and the recursive splitting algorithm is essentially a binary search, which has complexity on the order of $log_2 N$. The resulting complexity of the SM or IEPF algorithm is on the order of $N \times log_2 N$.

The incremental LT algorithm described in [44] involves starting with a line segment through the first two points in the scan, then adding new points sequentially and computing the new line parameters. If each new point is statistically likely to be a part of the line segment, the line segment is extended. Otherwise, the point is removed from the segment, the line parameters are recomputed, and a new segment is started. Since line parameters are calculated for each new point, and once again at the end of each segment, the complexity of the algorithm is $S \times N^2$, where S is the number of segments and N is the number of points in the scan. This method was evaluated in [44] to be slightly less than half as fast as the SM algorithm when used on identical sets of scans consisting of 722 scan points each, with similar performance otherwise. During this research, a significant improvement was made to the LT algorithm which reduced the complexity to the order of N, resulting in a decrease in the number of calculations required. By using a recursive least squares method for the line estimation, each new point is added to the line estimate without the need to recompute the line parameters using the entire data set. The modified incremental approach, herein called *Polar Recursive Line Segmentation (PRLS)*, adds new points to the line segment and updates the estimate parameters much in the way a Kalman filter incorporates new measurements as they arrive. A similar recursive linear regression approach is presented in [65] and [63], however those methods perform the linear regression in slope-intercept form

and thus require extra consideration for lines whose slope is infinite or zero. In PRLS, the linear regression is done using the normal form of the line, so the orientation of the line does not produce a singularity in the parameters.

The general form of the linear regression is discussed first, followed by the implementation of the recursive formula. Equation 24 shows the normal (or polar) form for a line expressed in Cartesian coordinates. Polar coordinates may also be used, however the result is more complicated and computationally expensive as discussed in the original formulation of the problem in normal form [2], which did not take advantage of recursive solutions to the problem.

In the normal form, a line is represented by two parameters: the perpendicular distance from the origin to the line, ρ, and the angle measured to the perpendicular line, ϕ. Every point on the line (x_i,y_i) satisfies the equation for the line, given the two parameters. Although mathematically a line through the origin is indeterminate in this form, it is not physically possible for a laser scanner, which collects data at incremental angles, to produce data which has a fit that goes through the origin. Thus, the smallest possible value for ρ for a given laser scanner is approximately equal to scanner angular resolution.

$$\rho = x\cos\phi + y\sin\phi = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \tag{24}$$

Using the two parameters, ρ and ϕ, a nonlinear regression can be performed to estimate a line given a set of scan points. However, by performing a change of variables, the problem can be set up as a *linear* regression with two new parameters, (A,B). Equations 25 and 26 show the linear regression formulation, with the solution for the new parameters given by Equation 27. Once the parameters (A,B) are calculated, ρ and ϕ can then be calculated using Equation 28 if desired (although in practice it is best to avoid doing so when possible).

$$\begin{bmatrix} \rho \\ \vdots \\ \rho \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix} \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} = \mathbf{H} \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \tag{25}$$

$$\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix} \begin{bmatrix} \frac{\cos\phi}{\rho} \\ \frac{\sin\phi}{\rho} \end{bmatrix} = \mathbf{H} \begin{bmatrix} A \\ B \end{bmatrix} \tag{26}$$

$$\begin{bmatrix} A \\ B \end{bmatrix} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} y_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} y_i \end{bmatrix} \tag{27}$$

$$\rho = \left(A^2 + B^2\right)^{-1/2} \tag{28}$$

$$\phi = \arctan\left(\frac{B}{A}\right) \tag{29}$$

A set of laser scan measurements can thus be analyzed and a line segment through the data can be found. However, determining which points are likely to lie on the same line must be determined prior to batch processing as described above. Several approaches have been suggested in previous work, including using a distance test or using a sliding window to batch process groups of points to determine how to best group them before running a batch process on the resulting segment after each new point is added [5]. These approaches face difficulty in determining the appropriate distance thresholds to apply or require multiple processing of the same data sets as line segments are expanded to include new points. The PRLS algorithm, however, uses the cumulative squared error of the fit as a statistically significant threshold and performs the estimation recursively. Thus, the identification of segment end points occurs as the data is being processed rather than as a preprocessing step. The recursive algorithm begins by batch processing m consecutive points to determine the starting line parameters ρ and ϕ. Theoretically, at least two points are required to initialize the recursive algorithm, but in practice three to five initial points

helps to reduce the number of very short line segments. The perpendicular distance of each point from the line estimate is given by Equation 30.

$$d_i = x_i \cos(\phi) + y_i \sin(\phi) - \rho \tag{30}$$

which can be written in the following form to avoid having to perform trigonometric operations:

$$d_i^2 = \frac{(x_i A + y_i B - 1)^2}{A^2 + B^2} \tag{31}$$

Using the error defined above, each new point (x_i,y_i) can then be checked to see if it fits the current line estimate to within three standard deviations. If so, the point is incorporated to the current estimate as described below. First, the matrix $\mathbf{P}$ is defined as shown in Equation 32, based on the segment initialization points. Next, a new matrix $\mathbf{P}_{new}$ is calculated using the formula shown in Equation 33.

$$\mathbf{P} \triangleq (\mathbf{H}^T \mathbf{H}) = \begin{bmatrix} \sum_{i=1}^{m} x_i^2 & \sum_{i=1}^{m} x_i y_i \\ \sum_{i=1}^{m} x_i y_i & \sum_{i=1}^{m} y_i^2 \end{bmatrix} \tag{32}$$

$$\mathbf{P}_{new} = \mathbf{P}_{old} + \begin{bmatrix} x_i^2 & x_i y_i \\ x_i y_i & y_i^2 \end{bmatrix} \tag{33}$$

Finally, the new parameters (A,B) can be estimated using Equation 34:

$$\begin{bmatrix} A \\ B \end{bmatrix}_{new} = \begin{bmatrix} A \\ B \end{bmatrix}_{old} + \mathbf{K} \left(1 - \begin{bmatrix} x_i & y_i \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}_{old} \right) \tag{34}$$

where the Kalman gain, $\mathbf{K}$, is given by:

$$\mathbf{K} \triangleq (\mathbf{P}_{new})^{-1} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{35}$$

Figure 40 shows an example of how line segments are extracted from laser scan data, and Figure 41 shows PRLS segmentation of a scan taken during experimental testing on the first floor of the Montgomery-Knight building. Altogether, a set of 144 laser scans was collected using the Hokuyo laser scanner shown in Figure 36(c). Each scan consisted of an average of 467 points out of a total possible 682. The set of laser scans was used to compare the performance of the PRLS algorithm with the IEPF algorithm by counting the number of arithmetic operations and number of output segments for each algorithm. Both algorithms produced similar results in the accuracy and qualitative shape of the segmented output. However, PRLS required on average 22901 operations per scan ($\sim 49N$), while the IEPF algorithm required 278886 operations on average ($\sim 67N \log_2 N$). Thus, the PRLS algorithm required fewer calculations than the IEPF algorithm by more than an order of magnitude. Using the same settings for minimum segment length and range variance, the PRLS algorithm tended to split the scans into more segments, averaging 25.4 segments per scan versus 15.7 segments for IEPF.

4.3.2 Dilution of Precision

As mentioned above, a typical scanning laser rangefinder performs its measurements by panning a laser across the environment and sampling the return at fixed angular increments. The result is a series of 1-D range readings recorded at consecutive, known angles. The error associated with each range measurement is a function of many parameters, but only the affect of range to target on error can be estimated for an individual measurement during real-time operation. As a result, it is common practice to estimate the range error as a function of the measured range, neglecting all other sources of error [2, 43]. However,

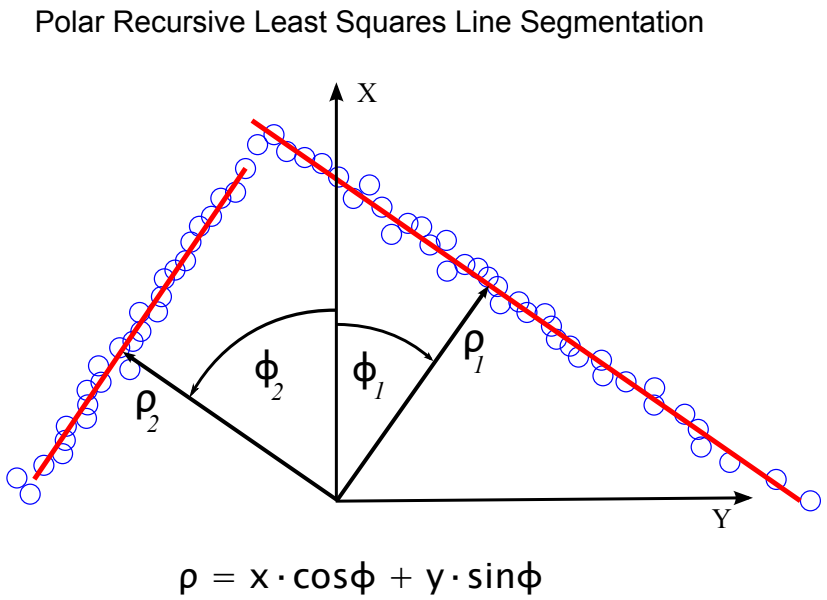


Figure 40: Line extraction using the PRLS algorithm. Each line segment is estimated from laser scan data using two parameters: perpendicular distance from the origin (ρ), and the angle to the perpendicular (ϕ).

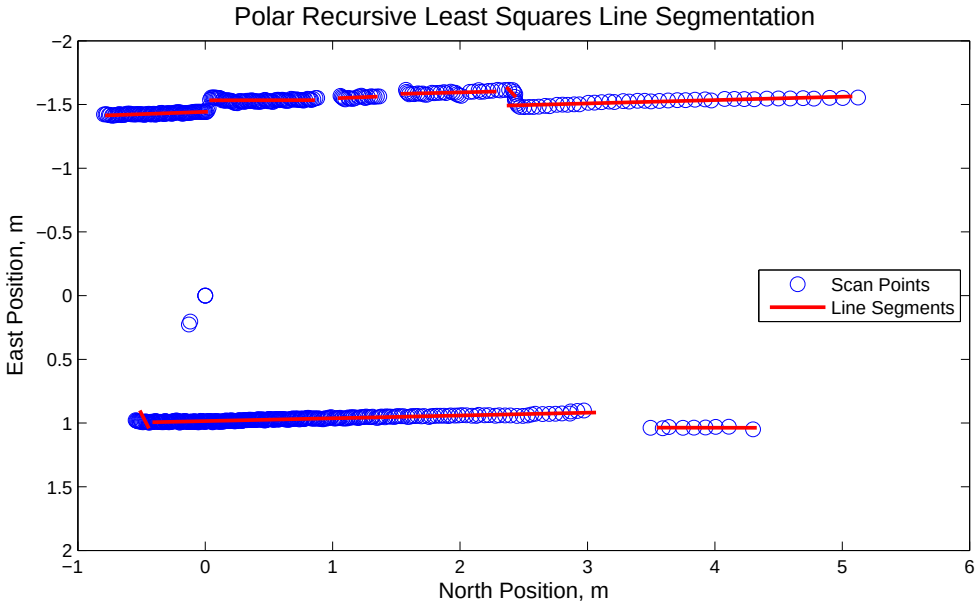


Figure 41: Line extraction using the PRLS algorithm on a laser scan collected during experimental testing.

if the local topology as indicated by the adjacent measurements is also considered, more information about the measurement uncertainty can be gleaned.

A similar problem exists for range measurements calculated using GPS signals. Termed Dilution of Precision, the effect of satellite position with respect to the receiver has an impact on the quality of the receiver position estimate. When the satellites used to estimate position are all located in a similar direction, the resultant navigation solution is less accurate. Similarly, the topology of the local environment measured by the laser scanner affects the accuracy of the pose estimate performed when matching scans against maps or other scans. In essence, vehicle position information is most accurate when measured perpendicular to the local environment. As such, a long featureless wall or hallway only provides position information perpendicular to the walls, and no information parallel to the walls. Thus, any position estimate using scans in this environment will show a strong correlation between the longitudinal and lateral estimates. This correlation is not detected by scan matching routines alone–the shape of the environment must be analyzed. Likewise, a heading estimate requires that range readings be different in different directions such that any change in heading can be detected. Hence, straight walls provide a good basis for measuring heading, while concave curved surfaces do not.

The information contained in a laser scan can be calculated by summing the information provided by each measurement to form the *information matrix*. An inverse form of the Kalman filter, called the *information filter*, utilizes the information matrix as defined in Equation 36 [18]. In this formulation, the *information* contained in each measurement is the inverse of the variance of the measurement, transformed by a measurement matrix, $\mathbf{H}$. In this case, the measurement matrix projects the variance, which is in the range direction, onto the vector normal to the surface. Then, the (x,y) components are calculated to get the information in each direction. For the heading component, it is noted that the range measurement has more heading information when the surface is close to parallel and farther away from the laser source, with the heading in the body frame with respect to walls (ψ) being the same as the direction to the normal (ϕ). The geometry of the problem is shown in Figure 42, with the measurement matrix defined in Equation 37.

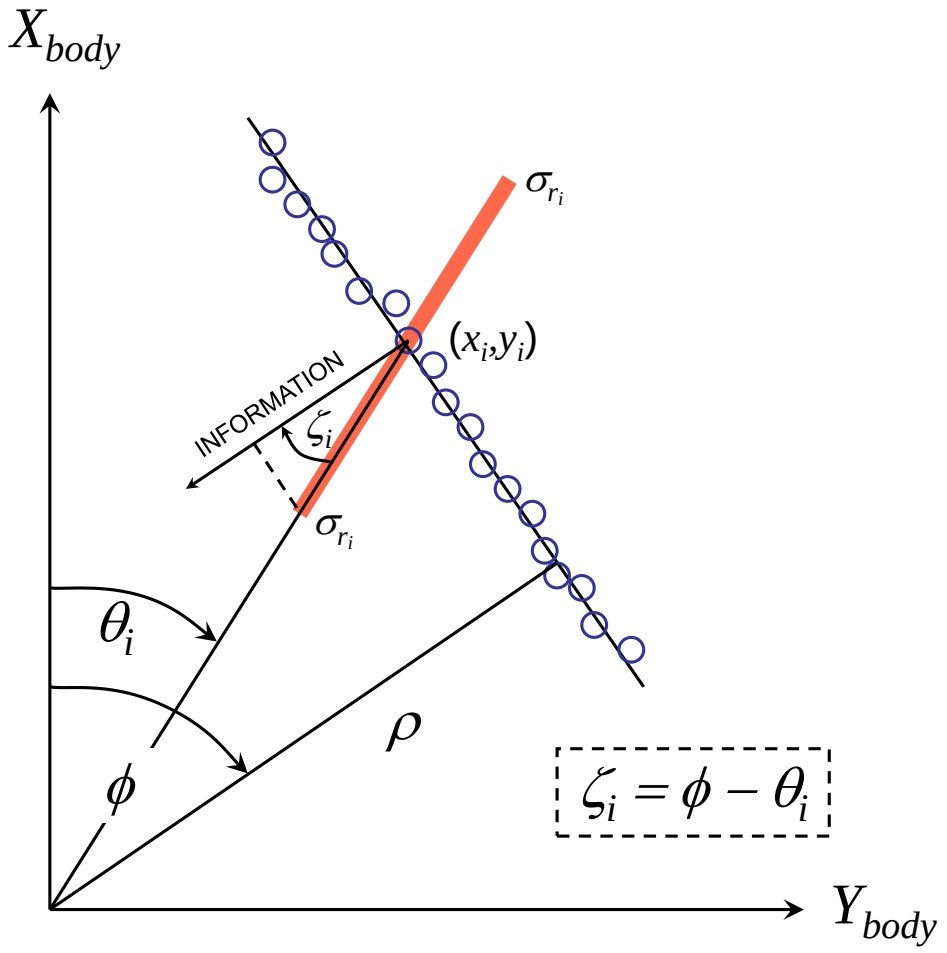


Figure 42: The information available from a laser scan is in the direction perpendicular to the local topology. The range error at each measurement angle is projected onto the information vector direction. This value is calculated for each of the scan points identified during the ICP scan matching algorithm.

$$\mathbf{I} \triangleq \mathbf{H}^T \mathbf{\Sigma_r^{2}}^{-1} \mathbf{H} = \mathbf{R}^{-1} \tag{36}$$

$$\mathbf{H} = \begin{bmatrix} \cos(\phi)\cos(\phi_1 - \theta_1) & \sin(\phi_1)\cos(\phi_1 - \theta_1) & r_1 \sin(\phi_1 - \theta_1) \\ \vdots & \vdots & \vdots \\ \cos(\phi)\cos(\phi_m - \theta_m) & \sin(\phi_m)\cos(\phi_m - \theta_m) & r_m \sin(\phi_m - \theta_m) \end{bmatrix} \tag{37}$$

$$\mathbf{\Sigma_r^{2}} = diag(\sigma_{r_1}^2, \sigma_{r_1}^2, \sigma_{r_1}^2, \ldots, \sigma_{r_i}^2, \sigma_{r_i}^2, \sigma_{r_i}^2, \ldots, \sigma_{r_m}^2, \sigma_{r_m}^2, \sigma_{r_m}^2) \tag{38}$$

The surface normal at each point is calculated by fitting a line through the measurement point and the two points on either side of it. The angle to the normal (ϕ_i) and the point coordinates (r_i,θ_i) are calculated for each range measurement. The result can be calculated as a sum of terms over the number of points in the correspondence set, m, when using ICP scan matching, or the total scan when using CoreSLAM. The equation for calculating the information matrix is given by 39, where the summation terms are defined in 40. Inverting the 3×3 information matrix produces the covariance matrix for the (Δx,Δy,$\Delta\theta$) pose error measurement, $\mathbf{R}$, which is used to update the EKF state estimate.

$$\mathbf{R} = \mathbf{I}^{-1} = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} & \Sigma_{x\psi} \\ \Sigma_{xy} & \Sigma_{yy} & \Sigma_{y\psi} \\ \Sigma_{x\psi} & \Sigma_{y\psi} & \Sigma_{\psi\psi} \end{bmatrix}^{-1} \tag{39}$$

$$\begin{aligned} \Sigma_{xx} &\triangleq \sum_{i=1}^{m} \frac{\cos^2(\phi)\cos^2(\phi - \theta_i)}{\sigma_{r_i}^2} & \Sigma_{xy} &\triangleq \sum_{i=1}^{m} \frac{\cos(\phi)\sin(\phi)\cos^2(\phi - \theta_i)}{\sigma_{r_i}^2} \\ \Sigma_{yy} &\triangleq \sum_{i=1}^{m} \frac{\sin^2(\phi)\cos^2(\phi - \theta_i)}{\sigma_{r_i}^2} & \Sigma_{x\psi} &\triangleq \sum_{i=1}^{m} \frac{\rho\cos(\phi)\cos(\phi - \theta_i)\sin(\phi - \theta_i)}{\sigma_{r_i}^2} \\ \Sigma_{\psi\psi} &\triangleq \sum_{i=1}^{m} \frac{\rho^2\sin^2(\phi - \theta_i)}{\sigma_{r_i}^2} & \Sigma_{y\psi} &\triangleq \sum_{i=1}^{m} \frac{\rho\sin(\phi)\cos(\phi - \theta_i)\sin(\phi - \theta_i)}{\sigma_{r_i}^2} \end{aligned} \tag{40}$$

While the above form preserves at least some geometric insight into the problem, the speed of the algorithm can be significantly improved in actual implementation by noting the identities in Equations 28 and 41 below. Substitution, and a little careful algebra, produces the form shown in Equation 42, thus enabling the calculation of the information matrix without having to resort to trigonometric operations. Figures 43-45 show the error ellipses associated with the pose measurement covariance calculated using the algorithm presented here using scan data collected during the experiment described at the end of this chapter. In the figures, the error is normalized using the sensor standard deviation and scaled up as indicated in the plots to improve visualization of the shape and relative size of the measurement uncertainty. The 1-σ position uncertainty is shown by an ellipse, while the 1-σ heading uncertainty is shown by a circle. In comparing the different scenarios, the relative size of the circle indicates the relative uncertainty in the heading estimates, while the size and shape of the error ellipse indicates the relative position uncertainty in the pose estimation.

$$\begin{aligned} \cos\phi &= \rho A & \cos(\phi-\theta_i) &= \cos\phi\cos\theta_i + \sin\phi\sin\theta_i \\ \sin\phi &= \rho B & \sin(\phi-\theta_i) &= \sin\phi\cos\theta_i - \cos\phi\sin\theta_i \\ \cos\theta_i &= \tfrac{x_i}{r_i} & \sin\theta_i &= \tfrac{y_i}{r_i} \end{aligned} \tag{41}$$

$$\begin{aligned} \Sigma_{xx} &\triangleq \sum_{i=1}^{m} \frac{A^2[A^2x_i^2 + 2ABx_iy_i + B^2y_i^2]}{C_i} & \Sigma_{xy} &\triangleq \sum_{i=1}^{m} \frac{AB[A^2x_i^2 + 2ABx_iy_i + B^2y_i^2]}{C_i} \\ \Sigma_{yy} &\triangleq \sum_{i=1}^{m} \frac{B^2[A^2x_i^2 + 2ABx_iy_i + B^2y_i^2]}{C_i} & \Sigma_{x\psi} &\triangleq \sum_{i=1}^{m} \frac{A[AB(x_i^2 - y_i^2) - x_iy_i(A^2 - B^2)]}{C_i} \\ \Sigma_{\psi\psi} &\triangleq \sum_{i=1}^{m} \frac{[B^2x_i^2 - 2ABx_iy_i + A^2y_i^2]}{C_i} & \Sigma_{y\psi} &\triangleq \sum_{i=1}^{m} \frac{B[AB(x_i^2 - y_i^2) - x_iy_i(A^2 - B^2)]}{C_i} \end{aligned}$$

$$C_i \triangleq (A^2 + B^2)^2 r_i^2 \sigma_{r_i}^2; \tag{42}$$

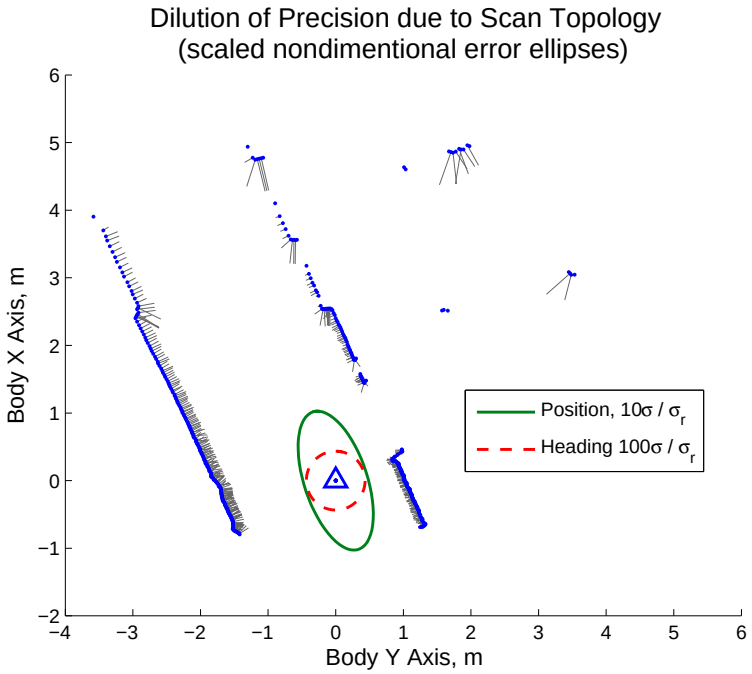


Figure 43: In a hallway, uncertainty is greater in the direction of the hallway.

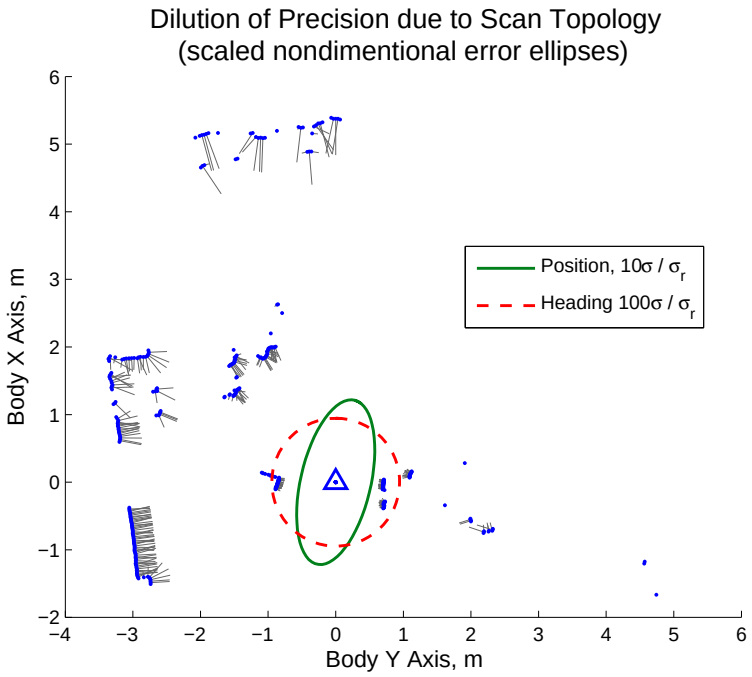


Figure 44: If many surface normals point toward the vehicle, the heading uncertainty is greater.

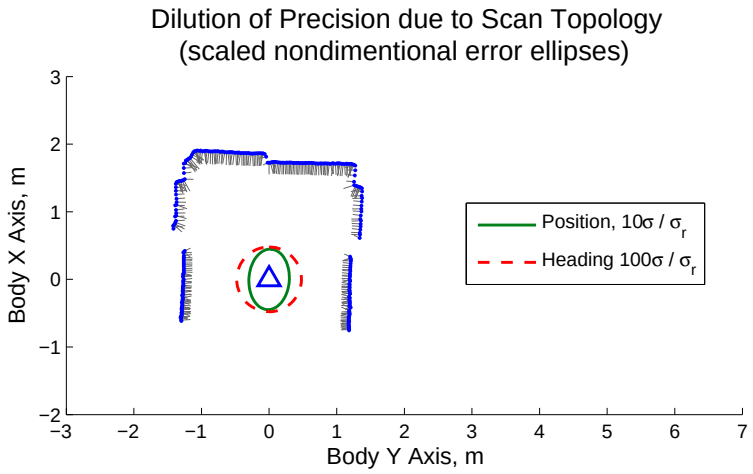


Figure 45: An environment with walls on several sides reduces uncertainty in the pose measurement.

4.4 Guidance Navigation and Control

4.4.1 Guidance Algorithm

Indoor navigation, as developed here, is by its nature based on relative measurements. As a result, any guidance system must use information in the local environment, rather than specifying waypoints fixed in inertial space. Thus, the existing path planning algorithm in the Georgia Tech UAV Research Facility guidance software, which uses inertially fixed waypoints, must be replaced with a relative guidance system. A simple velocity field method was employed, with a few basic rules for identifying different features of interest in the environment. In essence, the shape of the environment determines where the vehicle decides to go, and walls and doorways are identified and used in the decision making process. Each point in a laser scan contributes a small incremental velocity, and the average velocity over all the scan points is input as a commanded velocity to the velocity control loop. Equation 43 gives the equation used to calculate the incremental velocity at each scan angle [71].

$$\begin{cases} \gamma_i &= \theta_i + \frac{\pi}{2}(\frac{r_i}{r_{goal}}) - \pi & (0 \leq r_i \leq 2r_{goal}) \\ \gamma_i &= \theta_i & (2r_{goal} < r_i) \end{cases} \tag{43}$$

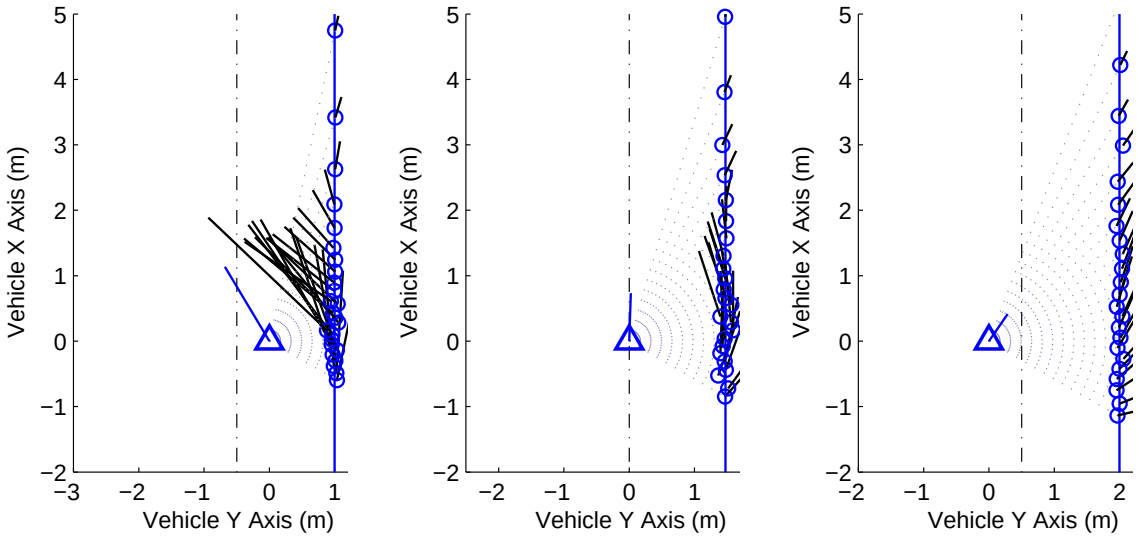
The incremental velocity at each measurement angle, γ_i, is determined by the range measurement, r_i, the range angle, θ_i and the user-defined desired distance from the wall, r_{goal}. The commanded velocity in the body frame is calculated according to Equation 44, where K_v is the velocity gain and n is the number of scan points with valid range readings. A user-defined minimum safe distance r_{min} (by default set to the vehicle maximum radius) is used to adaptively increase the gain K_v as a means of providing obstacle avoidance behavior as an inherent part of the velocity guidance algorithm. The body frame velocity commands are transformed to the local inertial frame using the vehicle estimated heading, ψ, as shown in Equation 45.

$$\begin{cases} x_{cmd_{body}} = \frac{K_v}{n} \sum_{i=1}^{n} \frac{cos(\gamma_i)}{|r_k - r_{min}|} \\ y_{cmd_{body}} = \frac{K_v}{n} \sum_{i=1}^{n} \frac{sin(\gamma_i)}{|r_k - r_{min}|} \end{cases} \tag{44}$$

$$\begin{bmatrix} x_{cmd} \\ y_{cmd} \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} x_{cmd_{body}} \\ y_{cmd_{body}} \end{bmatrix} \tag{45}$$

Heading control is achieved by using the angle parameters of the line segments extracted from the scan data using the PRLS algorithm. The vehicle is given a commanded heading to align itself with any portion of the wall set by the user. The default behavior is to use the farthest contiguous wall segment, starting from the beginning of the scan on the right and continuing counterclockwise until a gap is found in the wall that is large enough for the vehicle to fly through. Using this setting causes the vehicle to turn around when it encounters a "dead end" hallway while still making proper outside turns to go through doors or into side hallways.

The average of the incremental velocity commands provide the vehicle with a flight direction and speed that varies appropriately depending on the local environment topology. This guidance algorithm produces basic wall-following behavior, whereby the vehicle will

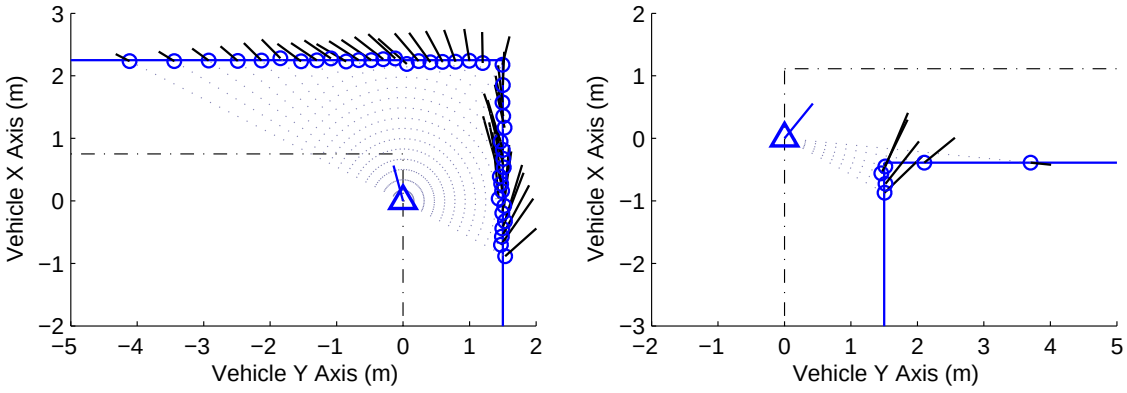


(a) Too close (b) Correct distance (c) Too far

Figure 46: The guidance algorithm generates a velocity vector from the average of vectors generated for each range measurement. Along a section of straight wall, the algorithm gives corrections to keep the vehicle a desired distance from the wall, which is indicated by a dotted line. The wall is shown by a solid vertical line, with individual range measurements shown by circles. Individual velocity increments are indicated by a line showing the magnitude and direction at each measurement point. The total commanded velocity is shown by a line emanating from the vehicle, which is represented by a triangle at the origin.

continue along a wall until it encounters an opening, an inside corner, or an outside corner. As the vehicle is flying along a wall, small changes in the velocity command keep the vehicle at the desired distance from the wall as shown in Figure 46.

When the vehicle encounters inside corners, no specific additional logic is required. The velocity increments generate a commanded velocity that slows the vehicle as it approaches the incoming corner, while at the same time the commanded velocity begins to turn in the direction of the wall ahead. At the same time, the heading command begins to track the oncoming wall instead of the current wall, so the vehicle also changes heading during the turn until it is flying along the new wall direction. For outside turns, the incremental velocity equation also automatically generates the correct velocity, and the heading algorithm provides a new heading command that is aligned with the corner. In both cases, the turn begins as soon as some portion of the next wall is visible to the scanner, and continues smoothly until the vehicle is completely following the new wall. Examples of inside and



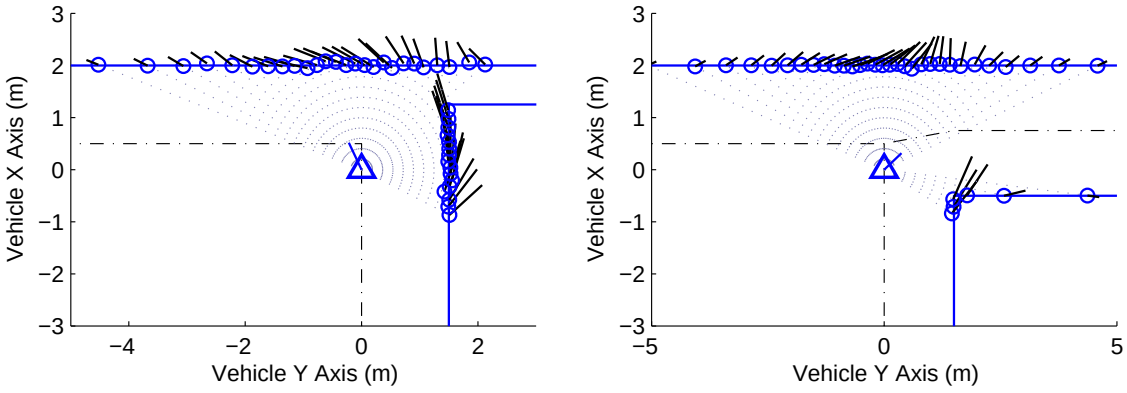
(a) Inside corner

(b) Outside corner

Figure 47: Inside and outside corners are handled with no special topology check required. The average velocity vectors generate a total commanded velocity that takes the vehicle smoothly through the maneuver. Heading commands are generated to turn the vehicle parallel with the next wall face as soon as it is detected.

outside turns are shown in Figure 47.

When a gap is encountered, either due to a physical discontinuity or failure of the sensor to detect portions of the environment, a small change is made to the incremental velocity formula shown above. Gaps are detected during the PRLS estimation as the scan is processed by looking for discontinuities in the segmented line estimate. If a gap is detected that is below a user-defined threshold, the guidance algorithm flags it as a "small" gap and it is ignored by the heading control portion of the guidance system (i.e. the heading algorithm treats wall segments on the far side of the gap as being connected to the rest of the wall). If the gap is an actual physical feature, any range measurements taken inside the gap will point toward the inside of the gap, causing the vehicle to fly slightly closer to the gap. When this occurs, any lateral incremental velocity command that is generated from points inside the gap are reversed such that they point outside the gap. This causes the vehicle to slightly *avoid* gaps, but the primary benefit is that it prevents the vehicle from becoming "stuck" against a gap that is just slightly smaller than the allowed entry size. In contrast, if the guidance algorithm detects a gap that is large enough to fly through, such as a side hallway, T-intersection, or open door, the horizontal incremental velocities are forced to all point *toward* the gap. This ensures that the vehicle will always fly into any opening that is large enough, which improves the overall coverage when the vehicle is



(a) Small gap (b) Large Gap

Figure 48: Any gap in the wall is detected by the PRLS wall estimation algorithm. Gaps that are smaller than the desired threshold are flown by with no effect on the commanded velocity. Gaps that are large enough for the vehicle to fly through, such as a hallway or door, cause the commanded velocity vector to bend toward the opening. If the vehicle enters a hallway or goes through a door where the minimum lateral separation is not at least twice the wall follow distance, the vehicle simply keeps an equal distance from both sides. Otherwise, the vehicle picks up the wall on the right hand side and keeps the desired distance from it as flight continues.

flying a wall-following search algorithm. In addition, the heading control routine will turn the vehicle toward passable gaps, improving the smoothness of the turn. Figure 48 shows an example of how gaps are managed by the guidance algorithm.

This simple guidance system is just one example of many algorithms that could be employed using the vehicle and navigation system described here. As the primary focus of this research is on navigation, only a simple algorithm that created velocity commands relative to the environment was required to demonstrate the effectiveness of the navigation system.

4.4.2 Navigation Algorithm

Vehicle state estimation for the navigation algorithm is based on the EKF design developed at the GT UAV Research Facility. The basic design is as described sections 2.1.2 and 2.2. In this implementation, the IMU measurements are incorporated at a fixed rate of 100Hz. In lieu of GPS position measurements and magnetometer heading measurements, the SLAM pose estimate and sonar range measurements are used, with updates at 10Hz and 20Hz respectively. The navigation filter estimates 15 vehicle states, $\hat{\mathbf{x}}(\mathbf{t})$, which are propagated

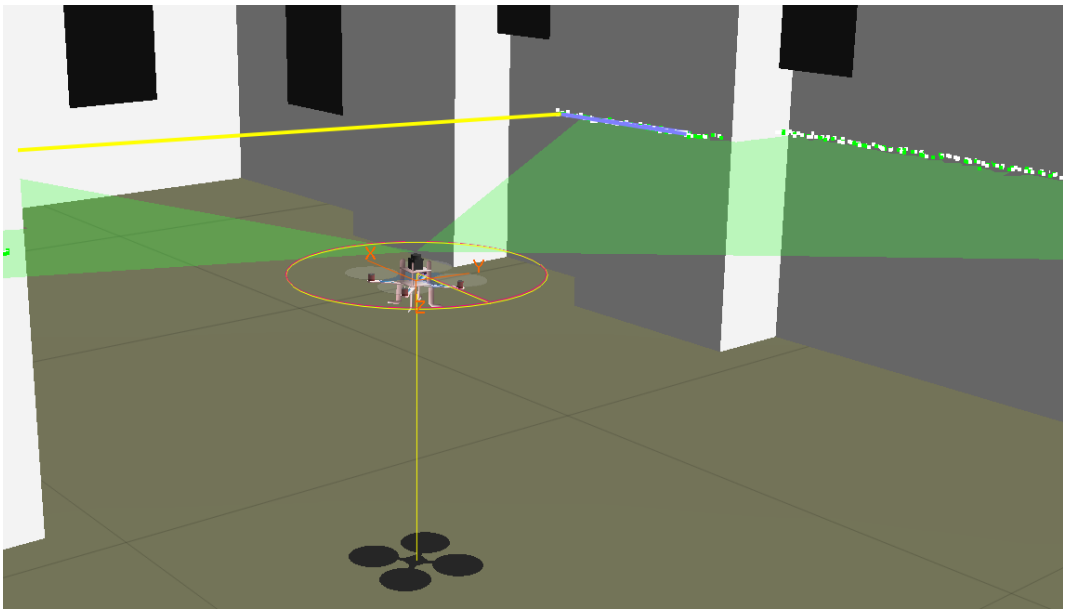


Figure 49: The guidance system uses the wall to the vehicle's right for heading alignment, and determines the location of the first possible doorway on the right. In the graphics display, the wall estimate is shown by a blue line, while the door location is spanned by a yellow line. Scan points are also visible here, with the key frame scan points indicated by white and the matched scan points from the current scan indicated by green.

using the linearized vehicle dynamics and updated using the sensor measurements with the appropriate covariance $\mathbf{R}_k$ and measurement matrix $\mathbf{H}_k$. The state and covariance propagation, as well as updates using the IMU measurements, are handled by the existing navigation software. Measurement updates from the SLAM pose estimation and sonar altimeter are incorporated by calling an external update function that takes as its arguments the measurement residual (sometimes called the "innovation"), the measurement matrix, and the covariance of the measurements. The measurement update equations given in Chapter 2 are repeated below as Equations 46-48.

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\big(\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-)\big) \tag{46}$$

$$\mathbf{P}_k^+ = \big(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k(\hat{\mathbf{x}}_k^-)\big)\mathbf{P}_k^- \tag{47}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\hat{\mathbf{x}}_k^-)\big(\mathbf{H}_k(\hat{\mathbf{x}}_k^-)\mathbf{P}_k^-\mathbf{H}_k^T(\hat{\mathbf{x}}_k^-) + \mathbf{R}_k\big)^{-1} \tag{48}$$

As shown below in Equation 49, the state estimate includes the vehicle angle error,

position, velocity, and biases of the IMU accelerometers and gyros. The process model, given by Equations 50-53 below, is a set of nonlinear differential equations describing the vehicle motion as described in detail in [28, 14]. The residual of the SLAM pose measurements (Equation 54) is the pose error (Δx, Δy, $\Delta\theta$), which is expressed in the inertial frame. The altitude residual (Equation 55) is simply the difference between the sonar measurement and the altitude estimate. It is assumed that the vehicle attitude does not affect the altitude measurement due to the properties of the sensor, as described in Section 3.1.3.2. The state and covariance updates occur at different times due to the different measurement rates of the SLAM navigation and the sonar. The sonar measurement is uncorrelated to other measurements, so its experimentally determined variance is used directly in the Kalman gain (48), with a measurement matrix equal to unity. The covariance of the SLAM pose measurement, calculated by inverting the information matrix shown in Equation 36, is in the body frame. Hence, it must be transformed using the direction cosine matrix that rotates body frame measurements into the inertial frame.

$$\hat{\mathbf{x}}(\mathbf{t}) = [\phi_{err}, \theta_{err}, \psi_{err}, x_{pos}, y_{pos}, z_{pos}, u, v, w, b_{ax}, b_{ay}, b_{az}, b_{\omega x}, b_{\omega y}, b_{\omega z}]^T \tag{49}$$

$$\dot{\hat{\mathbf{x}}} = \hat{\mathbf{v}} \tag{50}$$

$$\dot{\hat{\mathbf{v}}} = a(\mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t})) \tag{51}$$

$$\dot{\hat{\mathbf{b}}}_{\mathbf{a}} = 0 \tag{52}$$

$$\dot{\hat{\mathbf{b}}}_{\omega} = 0 \tag{53}$$

$$\left(\mathbf{z}_{SLAM_k} - h(\hat{\mathbf{x}}_k^-)\right) = [\Delta x, \Delta y, \Delta\theta]^T \tag{54}$$

$$\left(\mathbf{z}_{sonar_k} - h(\hat{\mathbf{x}}_k^-)\right) = (sonar\_measurement_k - EKF\_alt\_estimate_k) \tag{55}$$

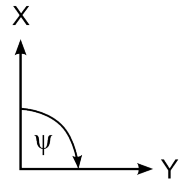
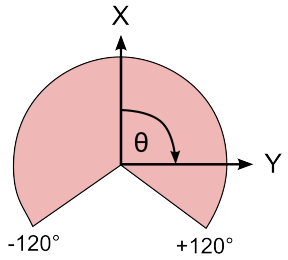
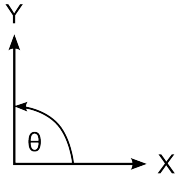
Care must be taken to calculate the residuals in a manner consistent with the navigation filter states. Table 2 Shows the different reference frames and their respective coordinate

systems used in the navigation algorithms. The simulation described below assumes a flat earth with a North-East-Down (NED) coordinate system. The navigation algorithm uses an inertial reference frame that is by default aligned with the simulation NED frame, although in practice the relationship between the navigation inertial frame and NED is defined by an arbitrary rotation about the Z-axis and a translation of the origin that is established when the navigation system is initialized.

The scan data is collected in a reference frame that is fixed to the vehicle body, with the X-axis aligned to the front of the vehicle, the Y-axis to the right, and the Z-axis down. The vehicle's navigation system maintains an estimate of the direction cosine matrix (DCM) relating the body frame to the inertial frame. The DCM is used to project body frame X- and Y-axes into the 2-D inertial XY-plane. This projection forms another reference frame, herein called the *2-D body frame*, which is located at the center of the body and has the Z-axis aligned with the inertial Z-axis, and the X-axis aligned with the heading of the vehicle. As laser data is collected from the sensor, it is projected into the 2-D body frame. This corrects for changes in range caused by roll and pitch of the vehicle, assuming that the surfaces detected are continuous in the vertical direction. Scan data in the 2-D body frame are then processed by the SLAM navigation algorithms and transformed to the navigation inertial frame to estimate the vehicle's motion. The CoreSLAM algorithm uses an additional reference frame, which is fixed in inertial space, and aligned with the X-axis to East, the Y-axis North, and the Z-axis upward. The map constructed by CoreSLAM uses this reference frame, the origin of which is shifted by a customizable offset such that the vehicle begins exploration at a user-defined location on the map. Figure 50 shows a consolidated view of the different coordinate frames in use.

During operation, the onboard software can be remotely configured to use either the CoreSLAM or the ICP navigation algorithms. Prior to takeoff, navigation initialization occurs relative to the environment with the vehicle body frame, the 2-D body frame, and the navigation inertial frame initially aligned. The vehicle inertial frame is not aligned to any external reference, such as a magnetic compass heading, nor is it aligned with any features in the environment. In fact, the navigation inertial frame will drift over time as

Table 2: Reference frames used for laser assisted inertial navigation.

Reference Frame	Coordinate System	Transformation
Navigation Inertial Frame		The inertial frame is aligned by default to the simulation North-East-Down frame. In practice, the inertial frame is aligned to the body frame on navigation system initialization. Note that this is a right-handed system, with the Z axis pointing into the page.
2-D Body Frame		The 2-D body fixed frame is related to the inertial frame by a rotation of the heading angle ψ about the Z axis. Scan data is projected into this frame before analysis. $\begin{bmatrix} X \\ Y \end{bmatrix}_{2D\ body} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}_{inertial}$
CoreSLAM Frame		The CoreSLAM frame is an inertially fixed frame that is oriented with respect to the navigation inertial frame by the following relationship: $\begin{bmatrix} X \\ Y \\ \theta \end{bmatrix}_{\substack{Core \\ SLAM}} = \begin{bmatrix} Y \\ X \\ -\theta \end{bmatrix}_{inertial} + \begin{bmatrix} X_o \\ Y_o \\ -\frac{\pi}{2} \end{bmatrix}$

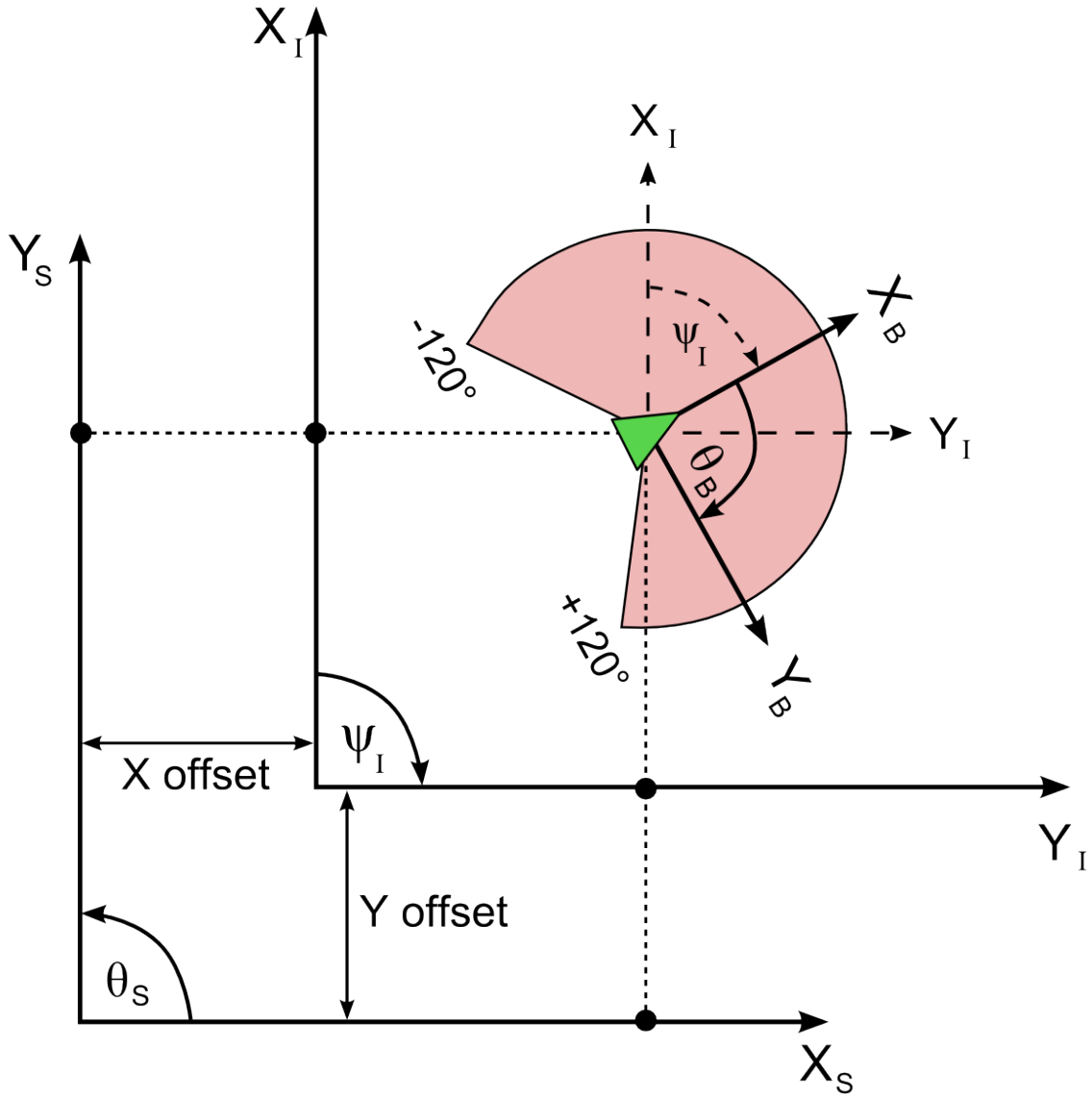


Figure 50: Coordinate frame relationships.

errors in the position estimate gradually build. These errors will cause distortions in a global map, but will not affect the vehicle's ability to navigate relative to the local environment or its ability to achieve position control with respect to the local environment, except when drift is in directions where there are no observable features.

4.4.3 Control Algorithm

The guidance and navigation techniques described above are compatible with control systems that have already been proven for unstable rotorcraft [27]. As a result, an existing adaptive control algorithm as described in Chapter 2 that utilizes a series of nested control loops (see Figure 2) was implemented for the simulation described below. The simulated guidance system provides the necessary position, velocity, and heading commands to the controller, and the navigation system provides the required state estimates for closed loop control.

4.5 Simulation of the Navigation System

4.5.1 Sensor Modeling and Simulation Environment

A laser scanner was modeled and the navigation algorithms described above were implemented using an in-house simulation environment developed at the Georgia Tech UAV Research Facility [31]. The simulation models the dynamics of various fixed-wing and rotorcraft vehicles, as well as sensors commonly available to many UAV systems including an IMU, a magnetometer, a sonar altimeter, and GPS. The simulation environment facilitates the development and testing of software that will later be flown onboard a vehicle. The *flight software* can be tested and refined by running it inside the simulation, with all sensor measurements and communications, vehicle dynamics, and wind gusts if desired, provided by the simulation. The system provides a graphic interface with access to system parameters that can be changed during flight and a variety of internal variables to aid in refining new simulated sensors and GNC algorithms. As is the standard in the Aerospace Engineering community, the simulation uses U.S. customary units since the flight software developed and tested in simulation is the same software that is flown onboard actual vehicles. Hence,

system parameters are listed below in U.S. customary units for the purpose of documenting actual settings used, while simulation and experimental results have been converted to metric units.

Before implementing the navigation algorithms, a model was developed for a scanning laser range sensor that could interact with the simulated environment. The simulation contains a database of objects which can be input by location, size, and object type. To build complex indoor scenarios, each wall is created by entering it at the command line or from an automated script. The scanning range sensor is "mounted" to a simulated vehicle, and it derives its position and orientation in the simulation from the parent vehicle. In order to detect the buildings, the laser scanner is modeled as a single ray that emanates from the desired sensor location on the vehicle. This ray is swept through a user-defined field-of-view, and the direction of the ray is calculated for each sensor step during the sweep. The ray direction is compared against each of the quadrilaterals in each of the walls to determine if a wall face has been hit by the ray. The distance to each wall face intersection is calculated, along with the incidence angle between the beam and the obstacle. The shortest distance detected is returned by the ray tracing function. If the closest obstacle detected is outside the user-defined maximum range, a "no obstacle detected" error code is generated for that range reading to mimic the behavior of the actual range sensor. Noise is added to the range reading using the mean and standard deviation provided by the user as a function of the range and incidence angle to the obstacle.

The sensor model is very flexible, with parameters that can be changed to meet the desired specifications of the user during runtime. Table 3 below shows the default settings, which correspond to the Hokuyo URG-04LX-UG01 range sensor. Noise characteristics were taken from characterization tests performed by Okubo, Ye, and Borenstein [47], and from the manufacturer's specification.

In the simulation, a semi-transparent green beam is displayed in the simulation window to indicate the location and orientation of the laser scan. The scanner range readings are updated at the desired scan rate and the measurements are stored in an array by the simulation software. Sensor communications are also simulated, such that a message that

Table 3: Default settings for the scanning laser range finder simulation.

Parameter	Setting	Description
Position	(0,0,0.4)	(X,Y,Z) Position relative to vehicle center of mass (ft)
Direction	(1,0,0)	(X,Y,Z) Orientation of sensor relative to vehicle (vector direction)
Bias	0 ft	Bias at zero reading
Sigma	0.015	% Standard Deviation, later multiplied by range measurement
Incidence Sigma	0.112 ft	Standard Deviation added at hight incidence angle
Maximum Range	18.37 ft (5.6 m)	Maximum sensor range
FOV	240°	Sensor field of view
Resolution	0.3515625°	Sensor angular resolution (360° /1024)
Scan dt	0.10 s	Scanner update rate

mimics the actual sensor is generated with the simulated range readings. The onboard flight software, running simultaneously, decodes the sensor message as if it were from the actual sensor. The flight software decodes the message and stores the range readings in polar and Cartesian coordinates for later processing by the guidance and navigation algorithms. A message containing the scan points converted to the inertial frame is sent from the flight software to the simulation for display to help the user visualize the scan measurement in realtime. Several other visualization aids are displayed in simulation, including the state estimate covariance (displayed as an error ellipse), the segment of wall used for heading control, and the location of doors detected. All of the visualization tools utilize data messages between the flight software and the simulation software, such that these visualizations would also be visible at a ground station terminal when the flight code is in operation onboard an autonomous vehicle. See Figures 38(a), 49 and 51 for examples of the simulation visualization window.

In addition to the sensor modeling and visualization tools described above, the baseline autopilot flight code was modified to include the navigation and guidance algorithms described above. No changes to the control architecture were required. Both CoreSLAM and the ICP algorithms were implemented as described above, with pose measurements fed into the existing EKF navigation system. The wall-following guidance system was also

implemented to test how well the navigation algorithms can determine position relative to the environment, and to see if the state estimates were accurate enough for the vehicle to fly stably and track relative position and velocity commands. In addition, vehicle guidance can be input manually, either using a joystick or by typing commands in the simulation console window.

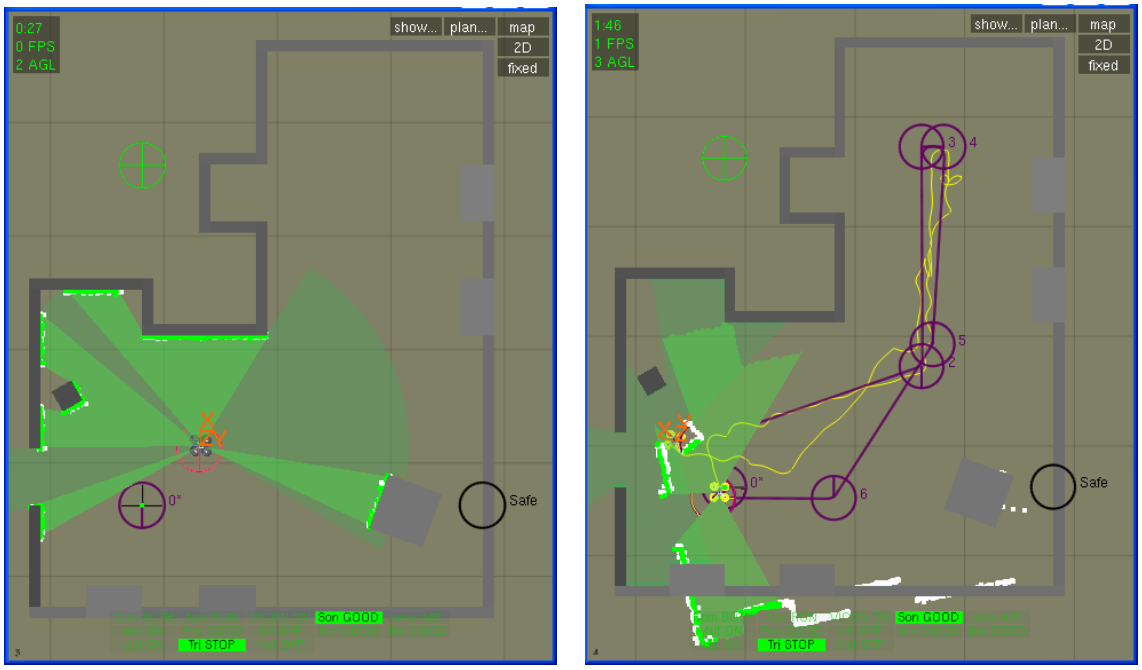
4.5.2 Simulation Results

The navigation and guidance algorithms described above were tested in a variety of different simulated indoor environments. A small quadrotor type aircraft, suitably sized for indoor flight, was selected for the vehicle dynamics model. Each test was performed using both the CoreSLAM and keyframe ICP navigation algorithms.

First, the vehicle was commanded to hover over a spot for three minutes. Next, a commanded trajectory was created using the simulation ground control station trajectory planning interface. The vehicle flew the commanded trajectory using only CoreSLAM or Keyframe ICP in lieu of GPS measurements for localization.

Using the SLAM position updates to the navigation filter, the vehicle is able to hover stably and fly a programmed flight plan through the simulated environment while relying entirely on the laser scanner for localization. Figure 51(a) shows the simulation environment and vehicle with laser scan displayed. The vehicle is shown hovering in the position hold test. The preprogrammed trajectory used in the second test is visible in Figure 51(b) as a curved path through the building with waypoints indicated by circles along the path.

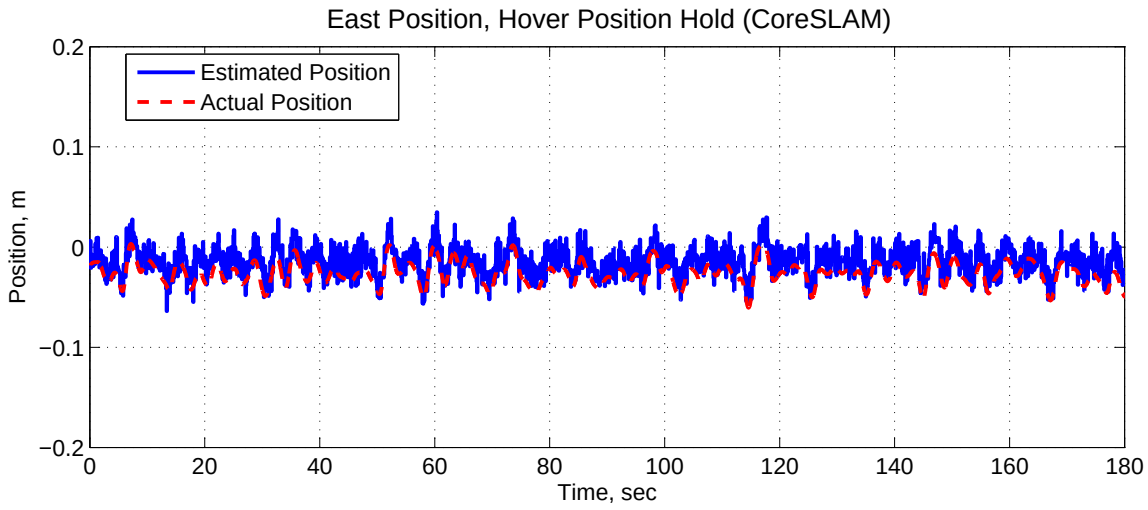
The actual and estimated position and heading during the hover maneuver are shown in Figures 52 through 54. Figure 55 shows the position error as the helicopter hovered around the commanded position. The vehicle was able to hold position better using CoreSLAM, although some drift was noticeable toward the end of the test. Using the Keyframe ICP algorithm, the position estimate was more accurate with no noticeable drift in spite of the increased oscillation around the hover point. Figure 56 shows the position estimate error when using CoreSLAM and Keyframe ICP to provide position updates to the EKF navigation filter.



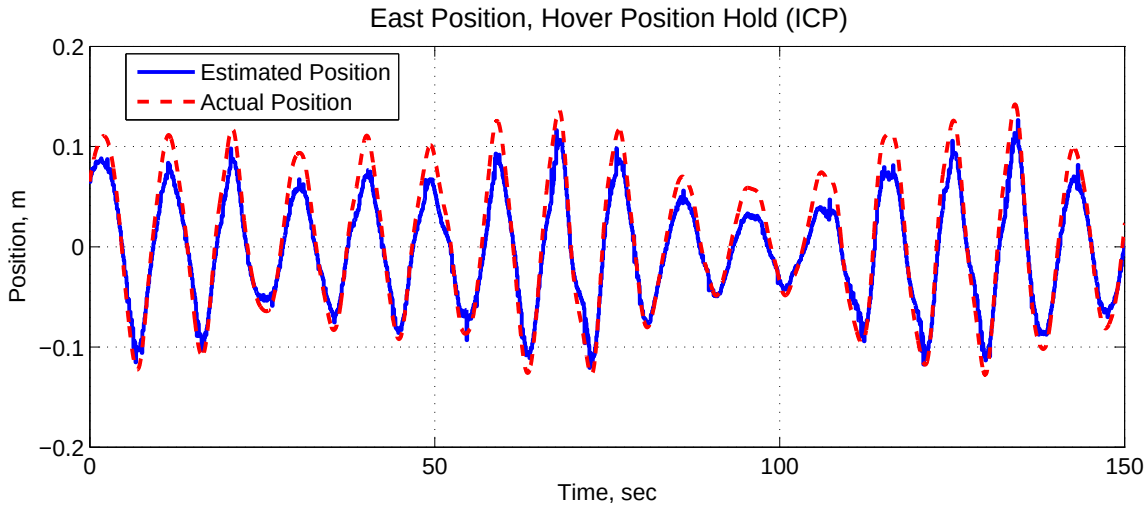
(a) Hover Position

(b) Trajectory Planner

Figure 51: The simulated position-hold hover maneuver was initiated at the location indicated in (a). Flight path waypoints are shown in (b) as circles connected by the commanded trajectory line segments. The final position and actual path after executing the commanded trajectory using the ICP algorithm is shown here.

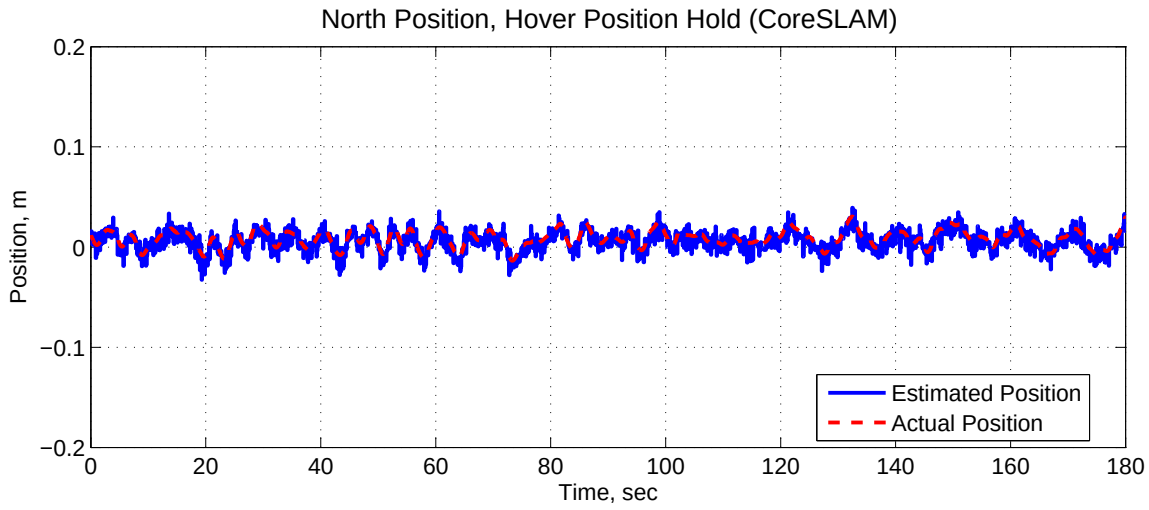


(a) CoreSLAM

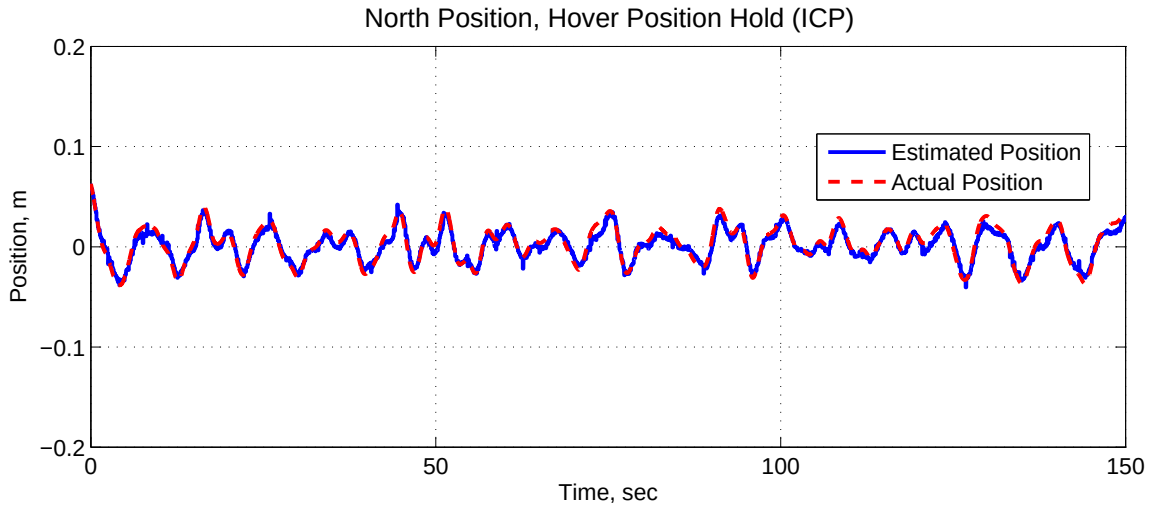


(b) Keyframe Scan Matching

Figure 52: East position during a three-minute hovering position hold maneuver using CoreSLAM and Keyframe Scan Matching for localization.

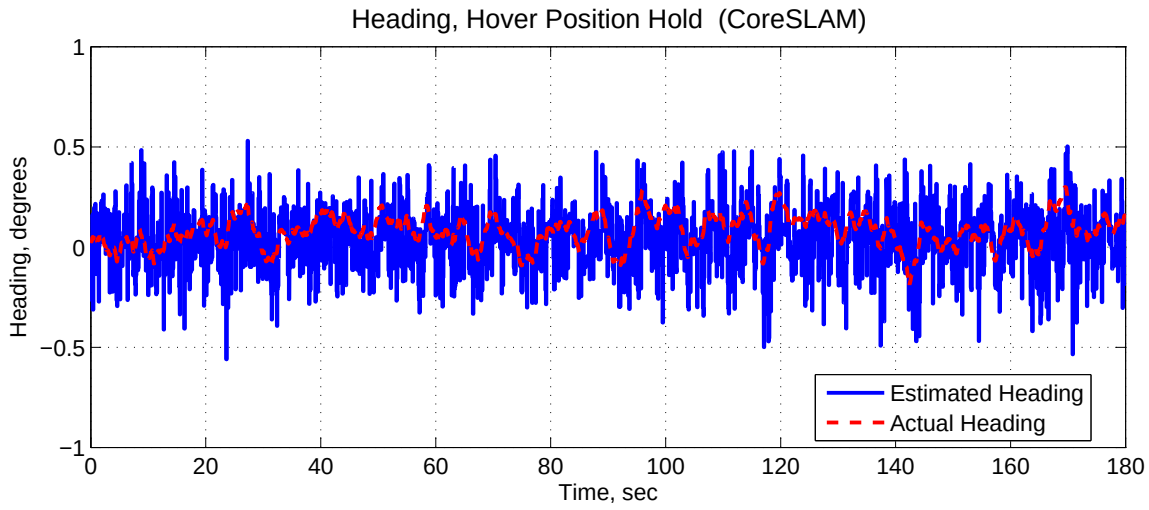


(a) CoreSLAM

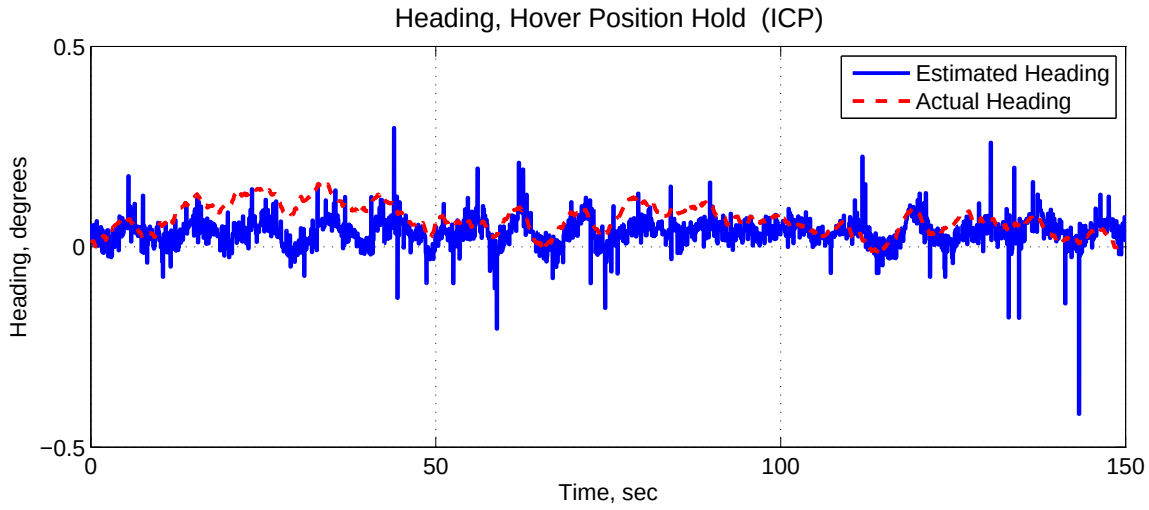


(b) Keyframe Scan Matching

Figure 53: North position during a three-minute hovering position hold maneuver using CoreSLAM and Keyframe Scan Matching for localization.

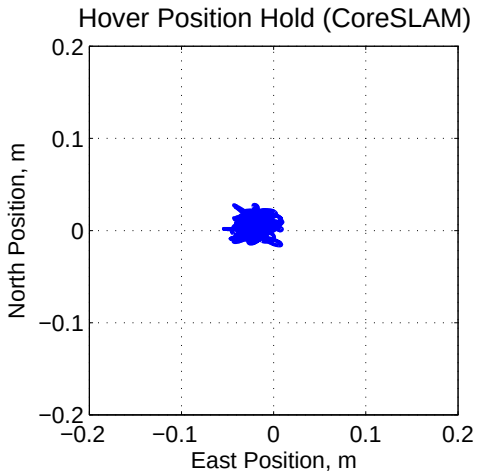


(a) CoreSLAM

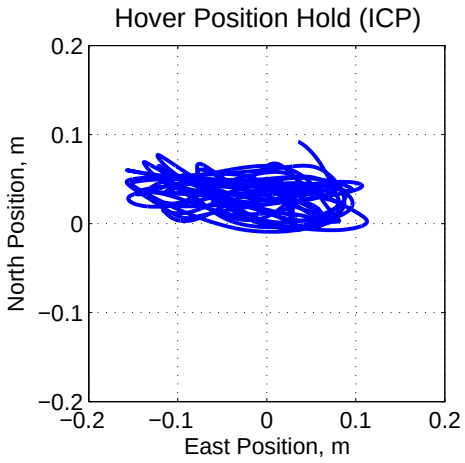


(b) Keyframe Scan Matching

Figure 54: Vehicle heading during a three-minute hovering position hold maneuver using CoreSLAM and Keyframe Scan Matching for localization.

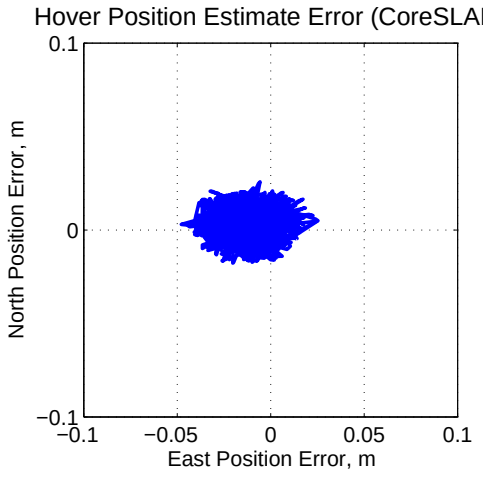


(a) CoreSLAM localization

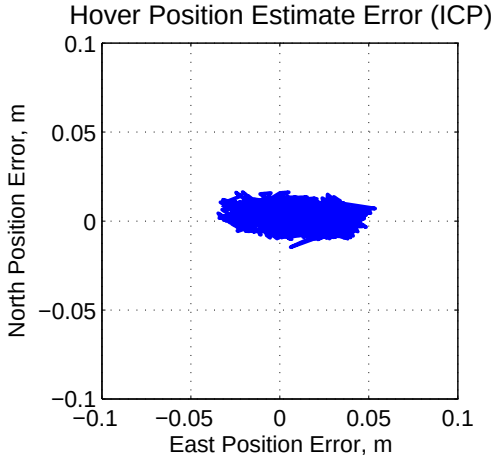


(b) Keyframe Scan Matching

Figure 55: Comparison of position error during a three-minute hovering position hold maneuver using CoreSLAM and Keyframe Scan Matching for localization. The commanded position is at (0,0).



(a) CoreSLAM localization



(b) Keyframe Scan Matching

Figure 56: Comparison of estimate error during a three-minute hovering position hold maneuver using CoreSLAM and Keyframe Scan Matching for localization.

After the hover test, the simulation was reset and the vehicle was allowed to complete its commanded trajectory through the building without interruption. Figures 57 through 59 show the position and heading errors during the tests using CoreSLAM and Keyframe ICP for localization. Both algorithms were successful, indicating that these SLAM algorithms can successfully replace GPS for navigation in indoor environments. Figure 60 shows the actual and estimated position, along with the commanded trajectory. During the Keyframe ICP test, key frames were updated as indicated by circles along the vehicle's estimated path. Using the default controller settings, the vehicle's autopilot seems to be able to control position better with CoreSLAM, however some adjustments to the controller may yield improvements and reduced oscillation using Keyframe ICP.

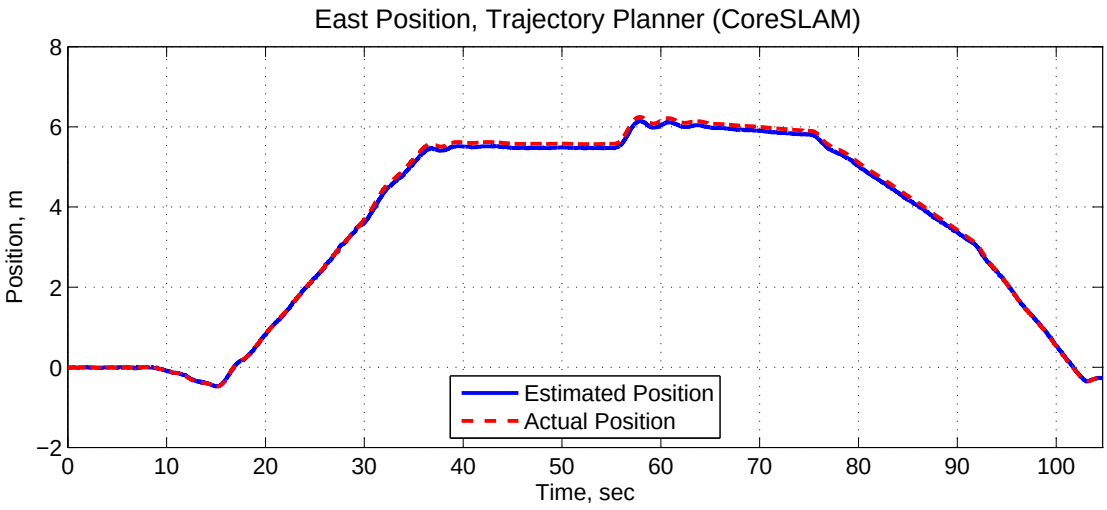
4.6 Experimental Results

An experimental vehicle is in the process of being designed and built to carry the sensors discussed above, so a test rig was assembled to compare the CoreSLAM and ICP algorithms in an experimental test. The test rig included a Hokuyo URG-04LX-UG01 laser scanner, an IMU made by Inertial Science, Inc., and a laptop computer to read the sensors, run the navigation software, display the ground station graphic interface, and record the data.

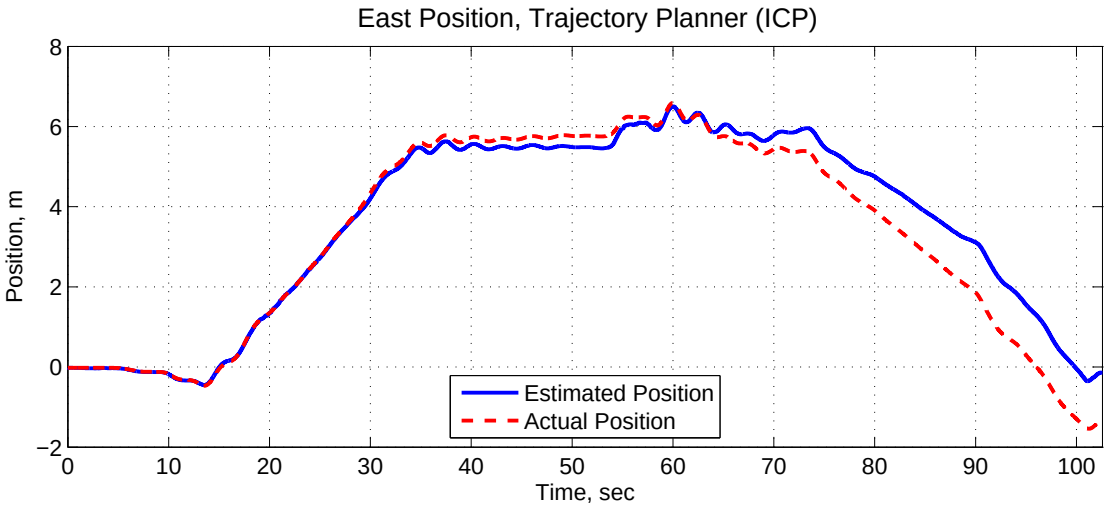
A long hallway was surveyed to determine the dimensions and location of prominent features, and the test rig was "flown" through the hallway to simulate vehicle motion and test the real time operation of the SLAM algorithms. The results of running the CoreSLAM and ICP algorithms are presented in this section.

4.6.1 CoreSLAM Scan Registration

The CoreSLAM localization and mapping algorithm was initialized using a map size of 400x400 pixels, with a map scale of 3. This resulted in a map size of 133.33 feet with a resolution of 4 inches per pixel. An initial offset of 60 feet in the South direction was used to shift the origin of the CoreSLAM map so that the entire hallway was visible on the map. The test rig was placed on a rolling cart and maneuvered down the hallway while the navigation filter maintained estimated position and heading using the IMU and laser scan data. Near the beginning of the test run, the sensor rig was diverted into a room to the

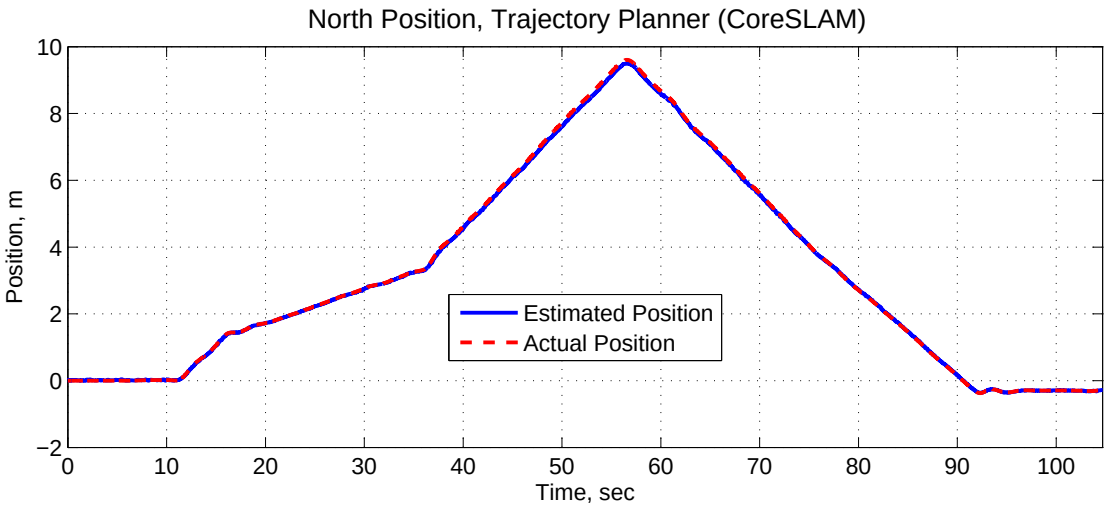


(a) CoreSLAM

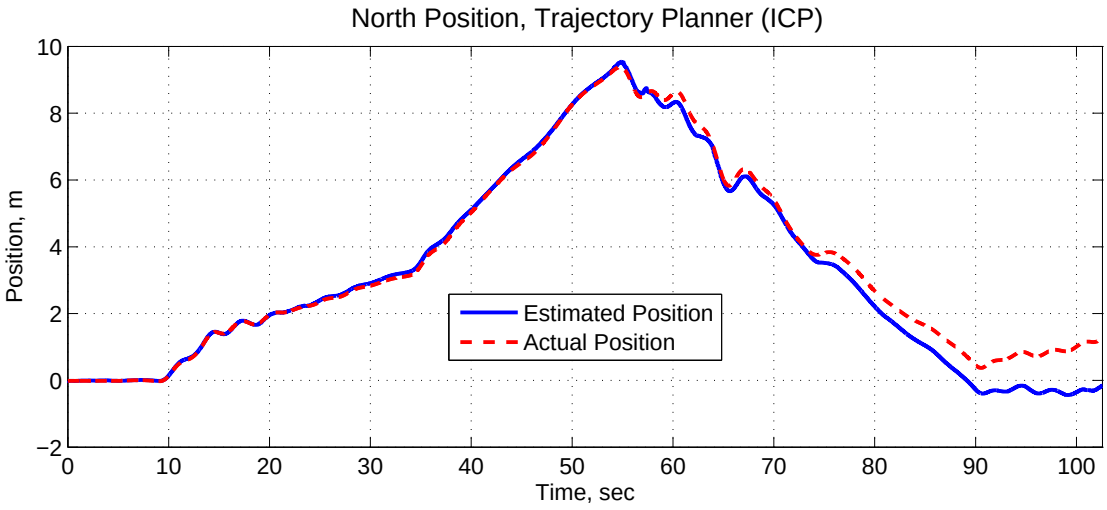


(b) Keyframe Scan Matching

Figure 57: East position while following a commanded trajectory using CoreSLAM and Keyframe Scan Matching for localization.

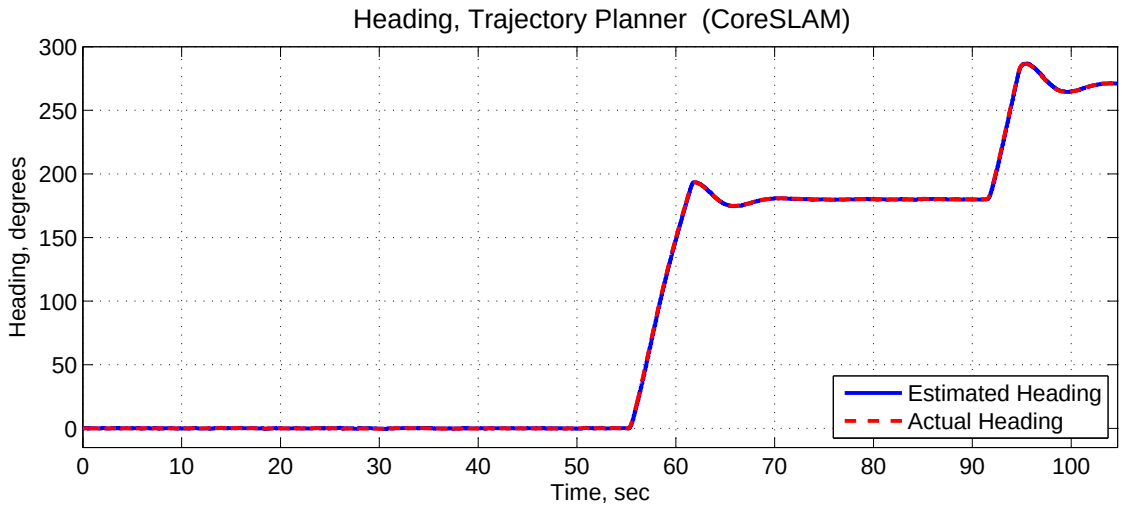


(a) CoreSLAM

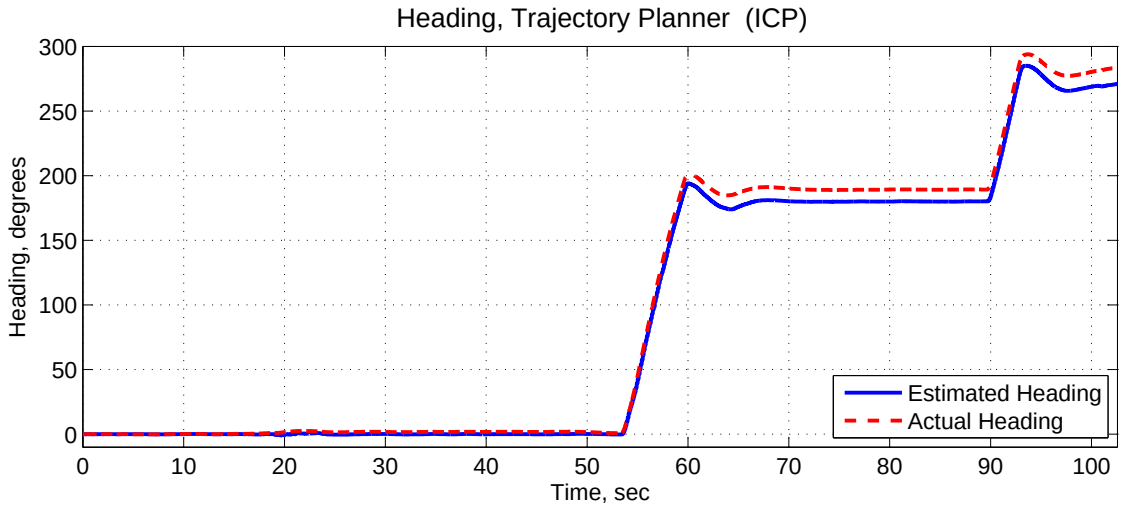


(b) Keyframe Scan Matching

Figure 58: North position while following a commanded trajectory using CoreSLAM and Keyframe Scan Matching for localization.

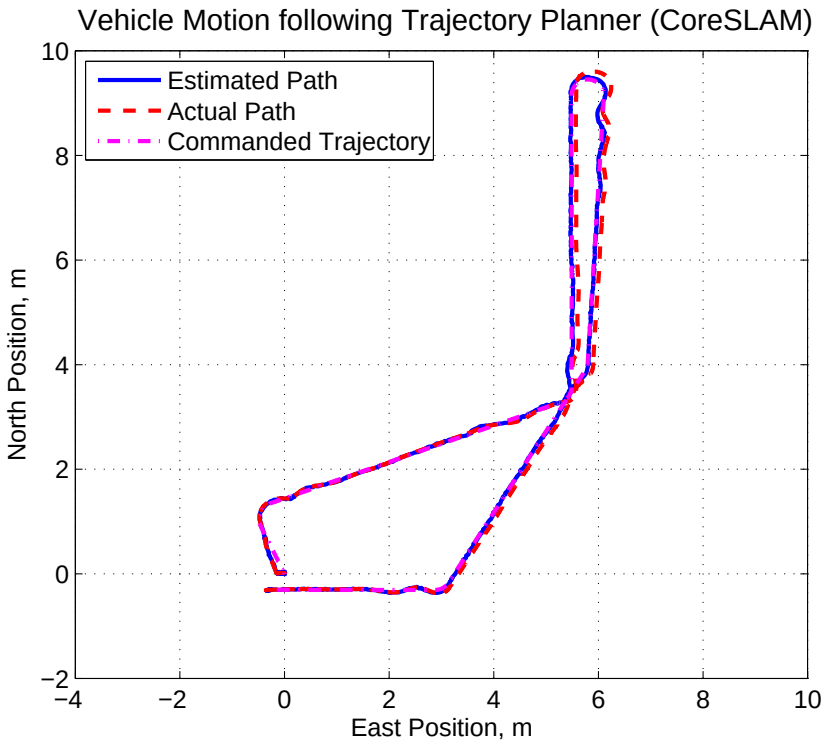


(a) CoreSLAM

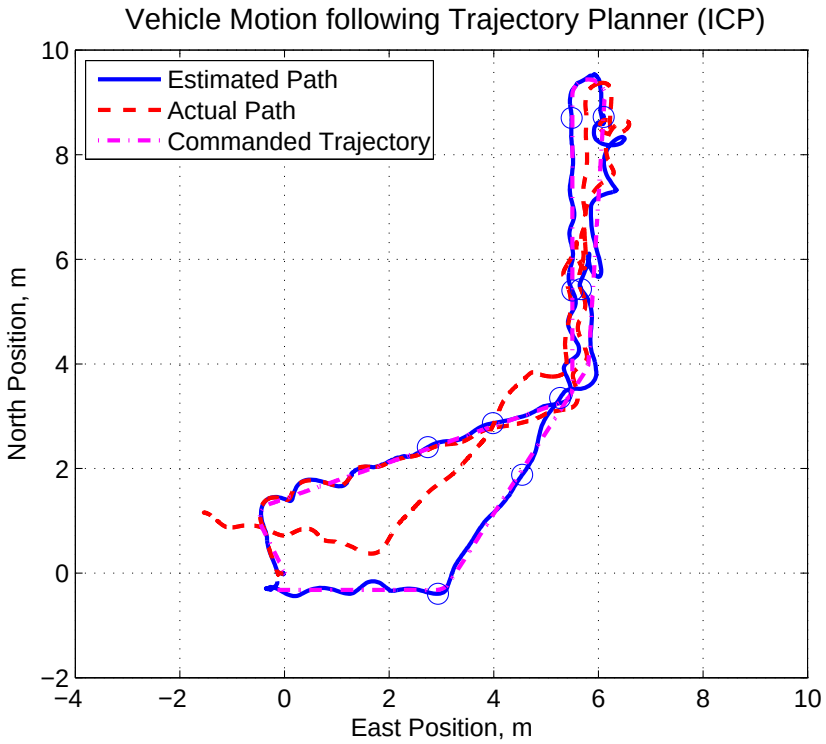


(b) Keyframe Scan Matching

Figure 59: Vehicle heading while following a commanded trajectory using CoreSLAM and Keyframe Scan Matching for localization.



(a) CoreSLAM



(b) Keyframe Scan Matching

Figure 60: Actual and estimated position while following a commanded trajectory using CoreSLAM and Keyframe Scan Matching for localization. In 60(b), the circles indicate where new keyframes were set along the path.

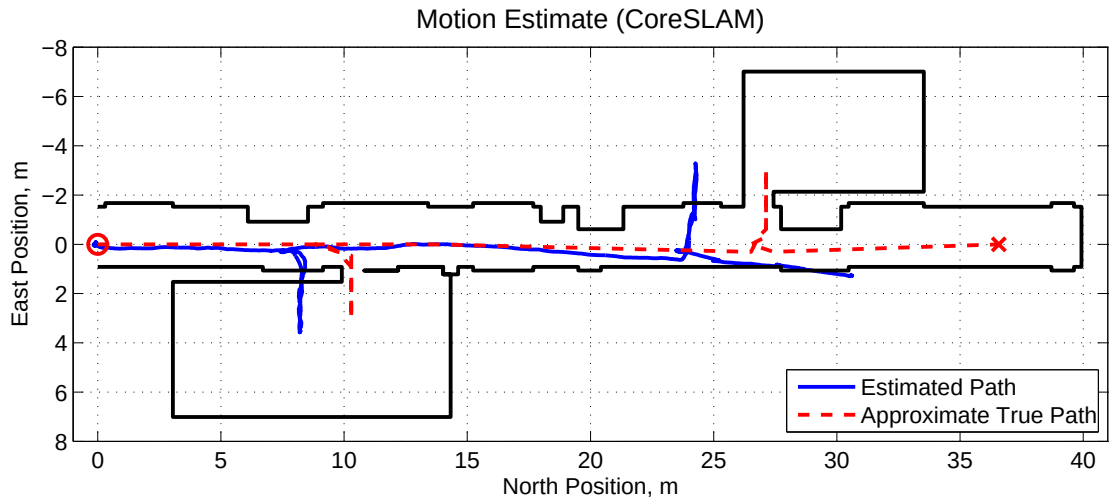


Figure 61: The motion estimate using CoreSLAM with an IMU. Wall location and approximate true path are shown for comparison.

East and then returned to the hallway. Toward the end of the test run, the rig was diverted into another room to the West, then returned to the hallway proceeding to the end of the hall. Figure 61 shows an outline of the test environment, an approximation of the actual path followed during the test, and the motion estimate created by the CoreSLAM-aided navigation filter. Figures 62 through 64 show the estimated position and heading during the test run.

The CoreSLAM algorithm position updates allowed the navigation filter to operate without significant drift throughout the test run. The majority of drift in this example is in the direction along the hallway. This is expected since there are fewer feature points along that direction and thus the scan registration is less accurate in that direction. Lateral and heading errors seem to be incurred primarily during the portions of the test when the rig is diverted into the side rooms. This may be a side-effect of using a rolling cart instead of a flying vehicle for the test rig. The center of rotation of the cart changes location depending on how the cart is maneuvered. This kinematic effect, along with the typical non-holonomic dynamics of rolling wheels, produces forces on the IMU that the navigation filter will incorrectly interpret using the dynamics of a flying vehicle.

Although these dynamic effects are unmodeled, the test setup demonstrated the ability of

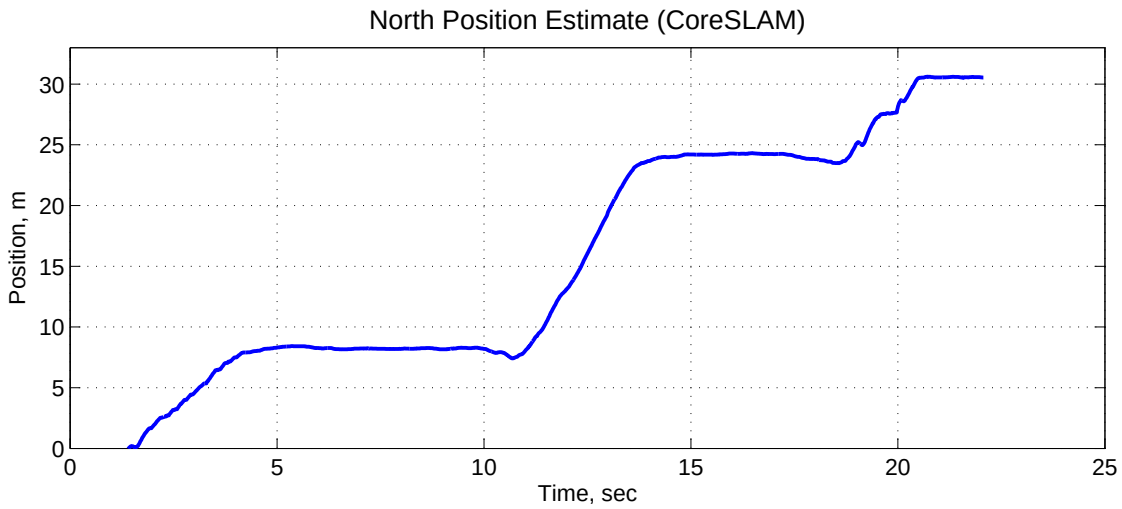


Figure 62: North position estimate using CoreSLAM scan registration and IMU. Note: time scale is compressed by a factor of approximately 4.

the laser-aided inertial navigation system to maintain a locally accurate position estimate with little drift over short distances. Figure 65 shows the map created during the test, along with the outline of the test environment for comparison. One notable effect observed during the CoreSLAM test was that the laptop computer was not able to process the laser scans in real time using even modest settings for the Monte Carlo scan registration routine. The navigation time increment is triggered by the external clock signals of the IMU at an assumed fixed rate. Since the computer was not able to run the algorithm in real time, the actual time stamp recorded on the data is inaccurate. As a result, the time scale in Figures 62 through 64 are compressed by a factor of approximately 4.

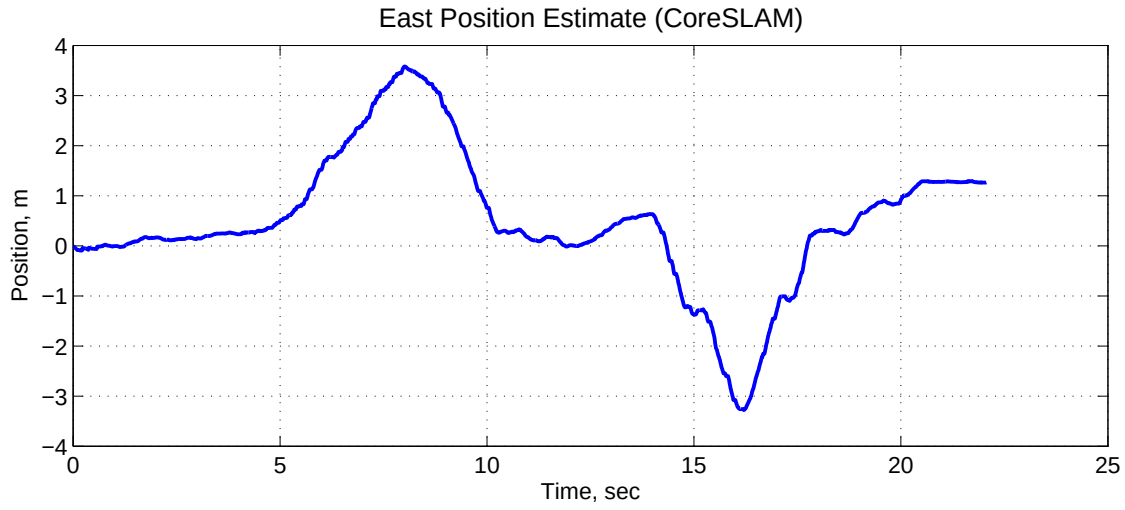


Figure 63: East position estimate using CoreSLAM scan registration and IMU. Note: time scale is compressed by a factor of approximately 4.

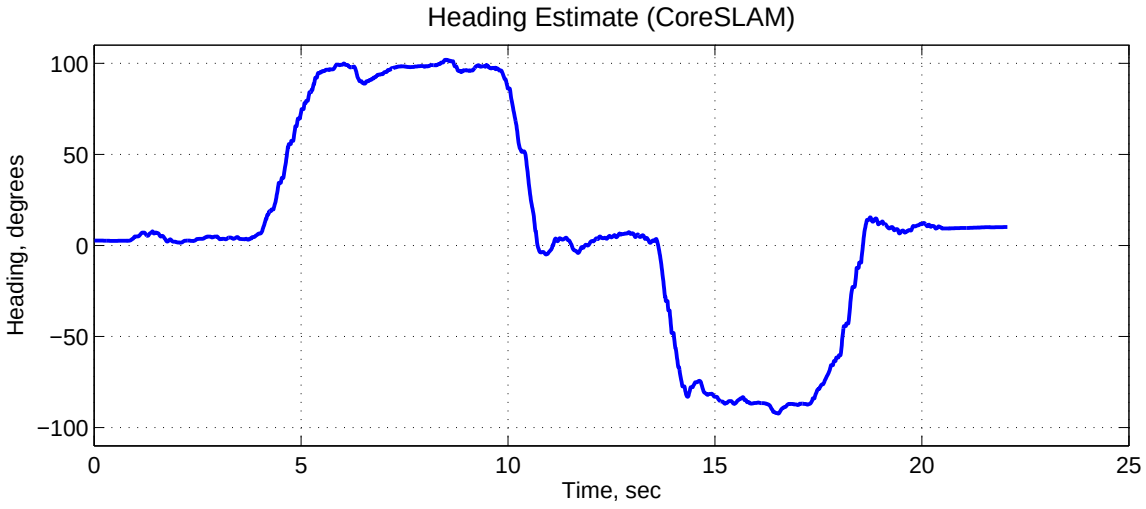


Figure 64: Heading estimate using CoreSLAM scan registration and IMU. Note: time scale is compressed by a factor of approximately 4.

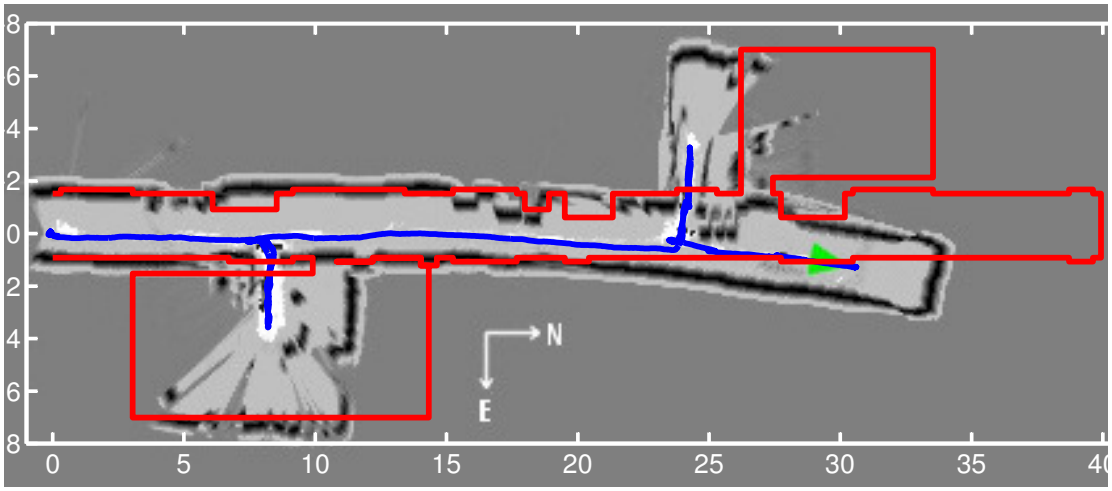


Figure 65: The map generated by the CoreSLAM algorithm with the motion estimate. Approximate wall location is shown by the solid red line.

4.6.2 Iterative Closest Point Scan Matching

The Keyframe ICP scan matching algorithm has several parameters that can be adjusted to improve speed and accuracy. The main parameters to set for the scan matching algorithm are the maximum distance between the key frame and current scan, the minimum number of corresponding points between scans, the maximum allowable error, and the maximum number of iterations. After some trial and error, a good balance was found to be a limit of 5 feet between key frame origin and current position estimate and a minimum of 250 (out of 682) scan points corresponding between the key frame scan and the current scan. If either of these two criteria were not met, the current scan was made the key frame scan and the process continued. A maximum squared-error of 0.01 ft^2 was chosen, so that the ICP match iterated until the error criteria was met or until the number of iterations reached 20.

Using the parameters described above for the scan matching loop, the same path was traversed with the experimental sensor rig through the test environment. Figure 66 shows the motion estimate created during the ICP test run. As seen with the CoreSLAM results, the motion estimate shows drift along the length of the hallway, where few feature points are available for scan matching. Lateral and heading errors are also visible, occurring primarily as a result of the turning maneuvers. As discussed previously, the motion of the rolling cart during the turns is expected to induce some errors due to the unmodeled dynamics of the system.

The ground station data recording was able to run the ICP algorithm at a maximum of 20 iterations per scan without lagging behind, showing a speed advantage over the CoreSLAM algorithm. However, the overall accuracy is somewhat less due to the fact that each time a new key frame scan is selected, any error up to that point is "locked in" so that the drift eventually grows as new key frames are selected. The North, East, and heading estimates for the ICP test are shown in Figures 67 through 69.

In addition to the pose estimates, it is interesting to note how the ICP algorithm performs at different points in the test run. The total number of iterations required, the error for each match, and the total number of correlated points at each successful match are shown in Figures 70 through 72. During the experiment, all key frame scans were logged

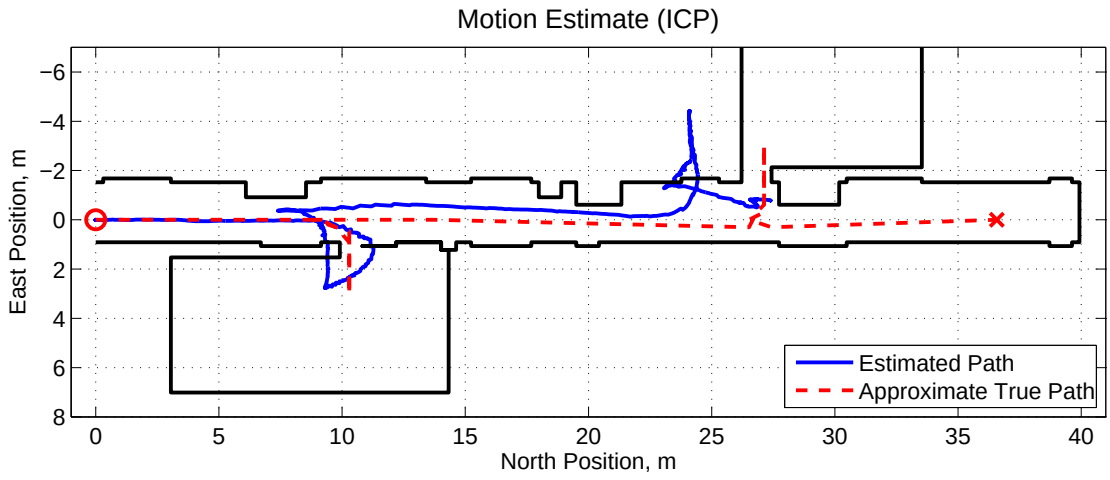


Figure 66: Vehicle motion estimate using ICP scan matching and IMU.

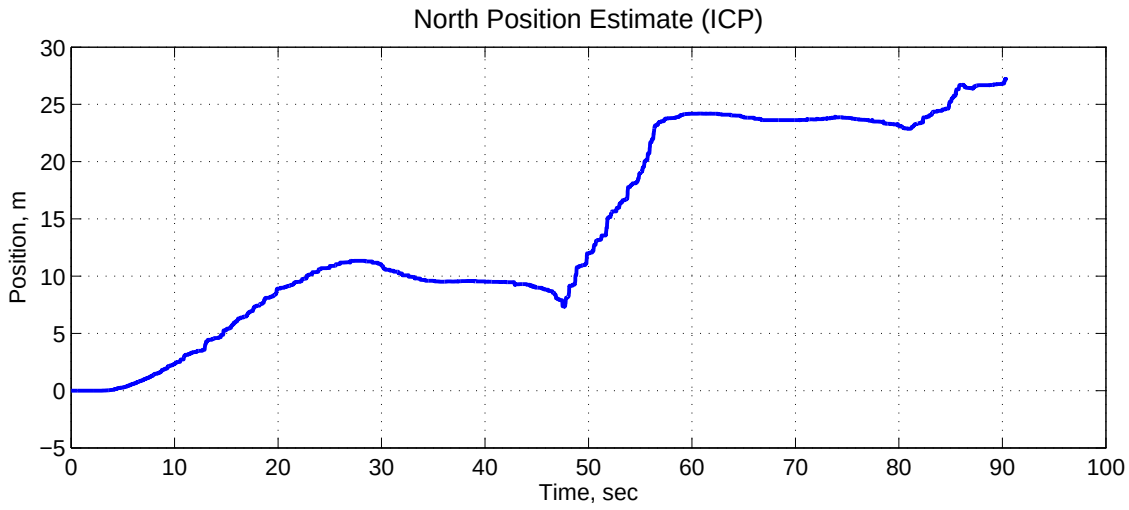


Figure 67: North position estimate using ICP scan matching and IMU.

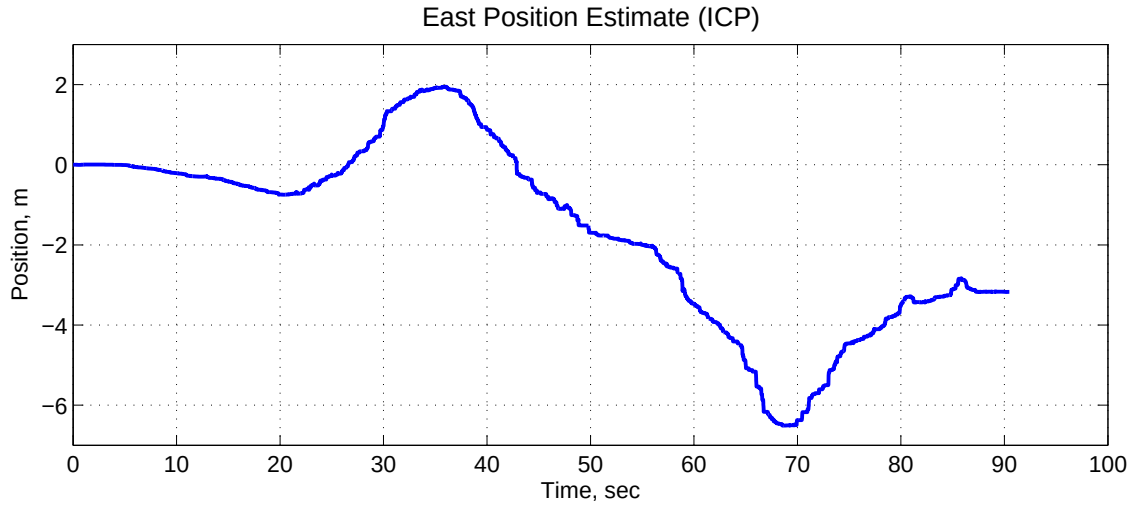


Figure 68: East position estimate using ICP scan matching and IMU.

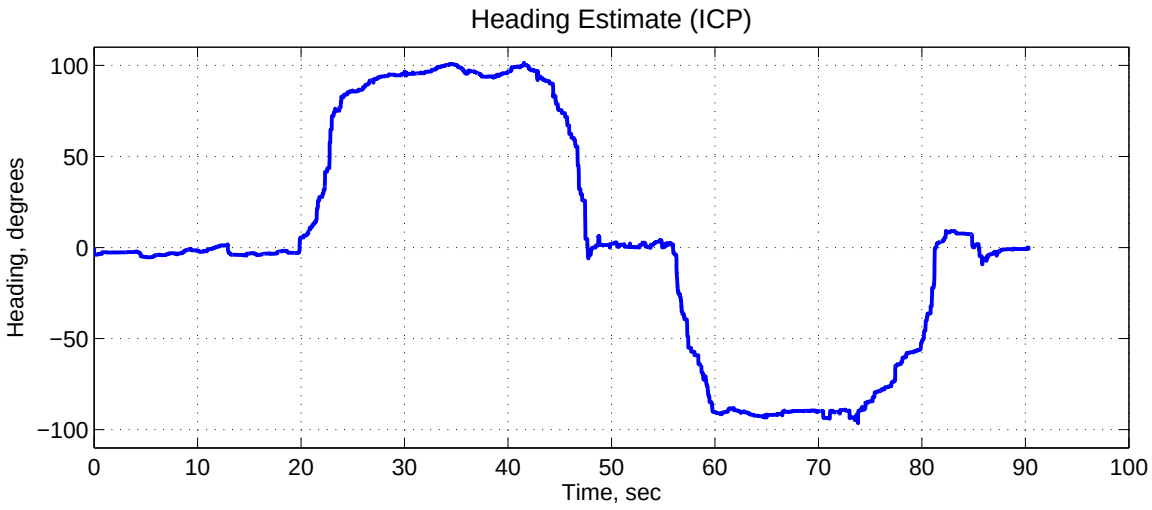


Figure 69: Heading estimate using ICP scan matching and IMU.

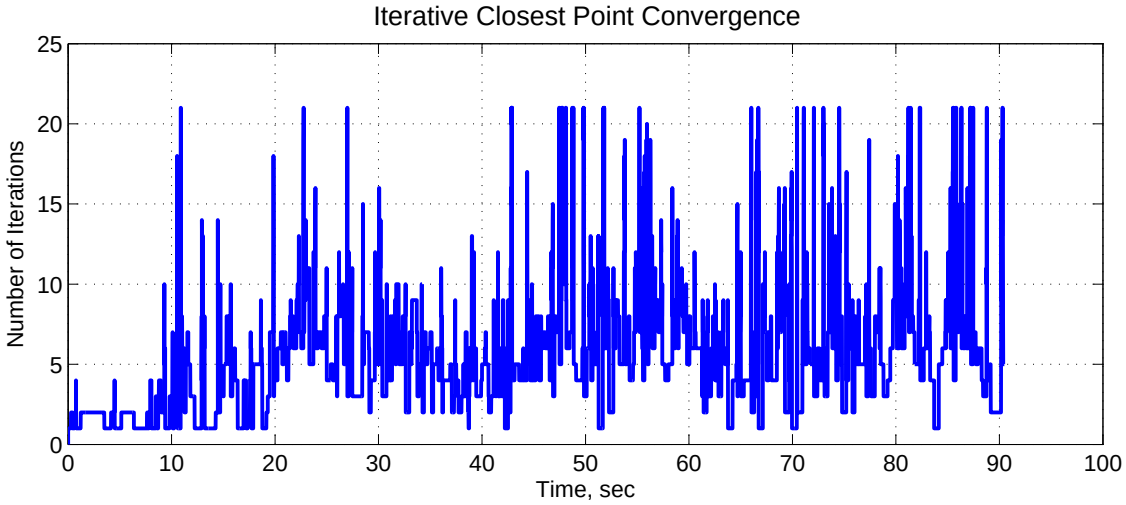


Figure 70: Number of iterations required during ICP scan matching routine.

and later they were combined into a "map" of the global environment. Figure 73 shows all of the key frame scans, along with the motion estimate and an outline of the actual test environment for comparison. Note that similar distortions occur as a result of errors in position and heading that accumulate during the test.

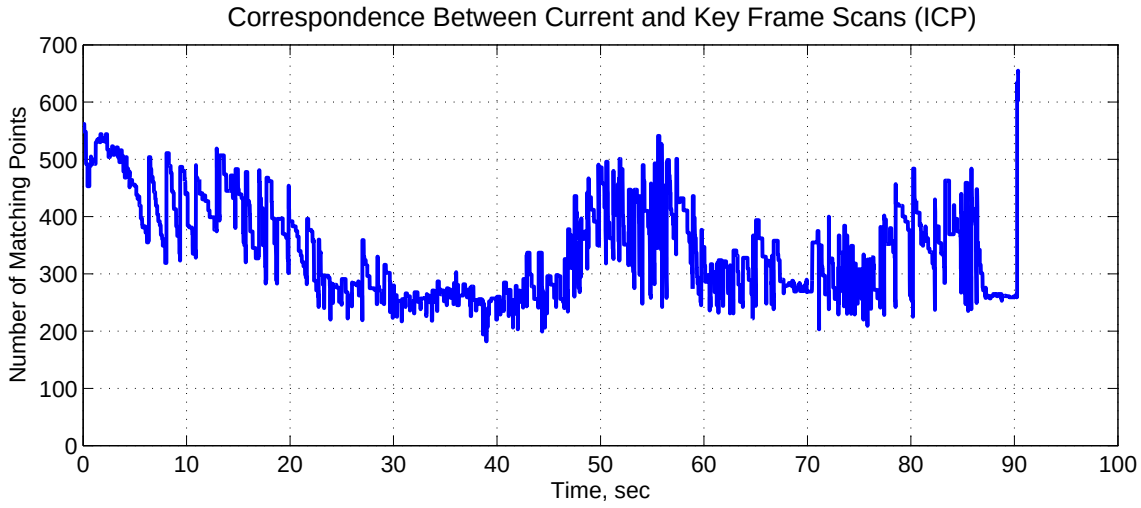


Figure 71: Number of correlated points between current scan and key frame scan using ICP scan matching with an IMU.

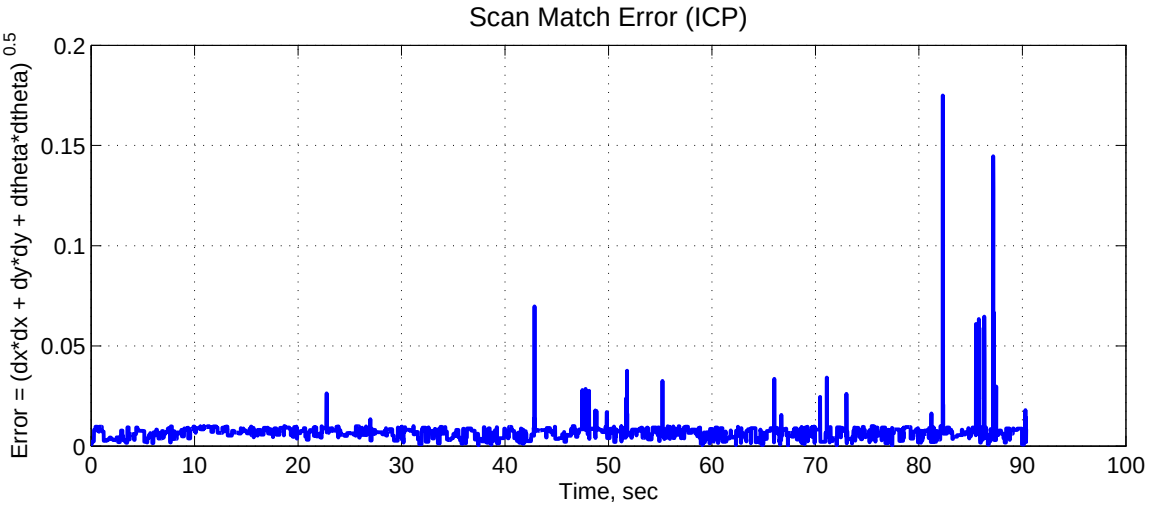


Figure 72: Error between current scan and key frame scan using ICP scan matching with an IMU.

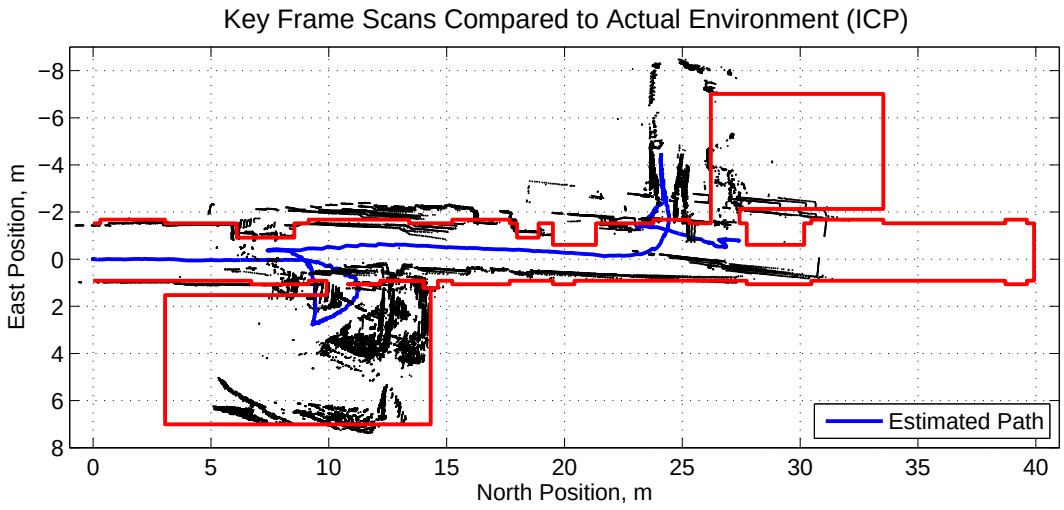


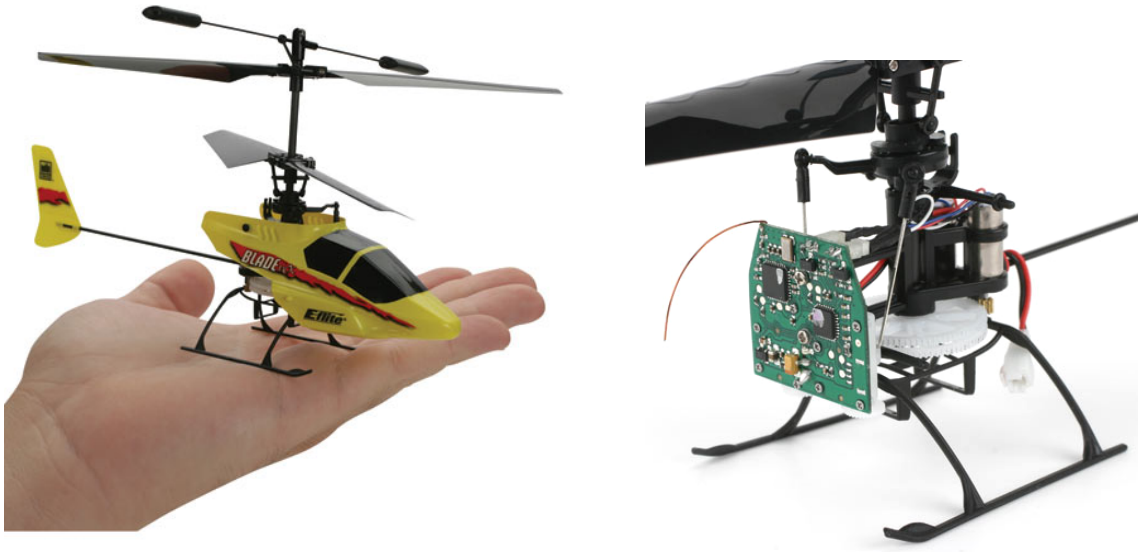
Figure 73: Key frame scans transformed to inertial frame, showing actual wall location for comparison (ICP scan matching with IMU).

CHAPTER V

CONCLUSIONS

This research provides several contributions in the field of autonomous flight vehicles, with an emphasis on solving problems related to indoor navigation. First, a vehicle capable of flying indoors autonomously using only five range sensors was designed, constructed, and successfully flown. This vehicle system was built using very low cost, lightweight sensors and a passively stable airframe. A sonar was used for altitude control, while IR sensors were used to measure range to walls, detect inside and outside corners, measure heading, and detect openings that the vehicle could pass through. Proportional-integral-derivative control was utilized to maintain desired distance to interior walls while employing lateral flight with wall-following behavior to explore an unknown indoor environment. The navigation, guidance, and control algorithms demonstrated during this research have direct application to miniature vehicles with a gross mass of less than 50g. Such vehicles could be built using current technology, with some small advancements in miniaturizing infrared sensor hardware. Figure 74 shows an example of such a vehicle, which has a stock mass of approximately 28g, that could be outfitted with miniature versions of the IR sensors described in this thesis. The GNC algorithms developed for this system can easily run on a small microcontroller chip, which would replace the stock radio control receiver circuitry.

The low-cost navigation system implemented on a passively stable vehicle performed well in the test setup described herein, and it is expected that performance would be comparable in similar environments. It is well-suited for uncluttered environments, and could likely descend stairs, although an ascending set of stairs would likely trigger the inside turning mode when the side-looking sensors detect the stairs during lateral flight. Obstacle avoidance in a cluttered environment would likely also cause a problem, since the IR sensors have a narrow beam width. Also, the current setup does not have any sensors aimed directly behind or above the vehicle, so these directions represent blind spots in the



(a) Blade® mCX by E-Flight®

(b) Blade® mCX without canopy

Figure 74: The navigation system developed and tested during this research has the potential for miniaturization and implementation on existing micro vehicles. This vehicle, manufactured by E-Flight®, has a stock mass of approximately 28g.

obstacle avoidance system. Operating in direct sunlight would also be problematic, due to increased noise on the IR range sensors, which are only rated for indoor use. In addition to interference from sunlight, outdoor flight would also be limited severely by wind due to the vehicle's limited maneuverability.

After demonstrating that low-cost relative navigation is feasible, more advanced navigation techniques were tested for suitability on an unstable rotorcraft vehicle. Two basic SLAM algorithms were developed and tested using simulation and actual hardware to determine the overall suitability of each for real-time autonomous indoor flight applications. An Extended Kalman Filter already in use for operational UAVs was adapted to incorporate measurements from a scanning laser rangefinder, via the SLAM algorithms, to estimate vehicle position in lieu of GPS. The first algorithm was a slightly modified version of an open-source algorithm called CoreSLAM. The second algorithm was a custom Iterative Closest Point scan matching routine that utilized key frame scans as a basis for localization. The use of key frame scans reduces the assimilation of position error over sequential scan matching methods. Both SLAM algorithms rely on a lean approach to the localization problem, demonstrating that methods compatible with current state of the art embedded

computers can be utilized.

A scanning laser rangefinder was modeled, and a simulated vehicle and environment were used to test the SLAM algorithms during autonomous indoor flight. Simulation results show that a laser scanner can be used to determine aircraft position in an indoor environment using these simple algorithms. Thus, the position drift that is inherent when using an IMU for stabilization can be corrected using the methods demonstrated without relying on GPS for localization. An experimental test rig was constructed to test both CoreSLAM and the ICP algorithm with an IMU in a realistic indoor environment. Both SLAM algorithms were able to provide position estimates to the navigation filter that were accurate enough to prevent short term drift, although some drift over long distances was observed. While the ICP algorithm was able to run in real time on a 2GHz laptop, processing scans as they arrived at 10Hz, the CoreSLAM algorithm was only able to run at approximately 25% of the required speed. Depending on the dynamics of a particular flying vehicle, position updates at that speed may be acceptable. In addition, an onboard computer will not be burdened with the additional task of running a graphic user interface and logging test data, and will likely have much less overhead than the laptop operating system in use during the test.

Both methods of navigation using laser scan data are limited to environments where features are detectable and static. If there are no observable physical features (for instance smooth, straight walls), or if obstacles in the environment are in motion, the two pose estimation algorithms tested here will likely perform poorly if at all. In addition, the 2-D nature of the laser scanner measurements could limit navigation performance in environments where small changes in altitude cause large changes in the scan topology. For instance, although the unstructured environment of a cave would provide a wealth of observable features, variation in the cross-section with altitude might make it difficult to perform scan-matching. The best type of environment for both algorithms would contain angular features that vary in size and distribution, as is common in manmade structures, with some regularity in the vertical dimension such that small changes in altitude do not result in large changes in the scan data.

The CoreSLAM algorithm, while very accurate, was also slower than the key frame ICP

method. Although actual implementation on an embedded computer may prove that real-time operation is feasible, especially as computer performance increases over time, there are methods for improving the algorithm that should be investigated. The mapping and scan-registration are already on the order of N, however the Monte Carlo pose estimation routine runs a fixed number of samples on every scan. An adaptive algorithm could be utilized, whereby the number of runs is either terminated early if a good match is found, or continued longer when needed.

The key frame ICP algorithm could also be extended and improved in several ways. First, the key frame scans could be stored in an array such that the history of key frames could be used to eventually solve the loop closure problem and correct large map errors if desired. Key frames could also be stored as a series of line segments instead of complete scans, thus creating a model of the environment. This would reduce the scan point correlation problem from the order of N^2 to the order of N, resulting in further improvements in speed and perhaps better pose estimation altogether.

In general, ICP scan matching underestimates heading changes, since rotation differences between scans usually result in a local minimum being found as the scans are rotated into place, but before the actual heading error is determined. Thus, during a turn heading error can accumulate, especially if a new key frame is selected during the maneuver. This could be resolved by placing more weight on the IMU measurements when yaw rates are high. Improvements in key frame transition could also be made by allowing scan matching to span across multiple key frames. Although this may be impractical when using the nearest neighbor scan correlation due to its computational complexity, if key frames are stored as line segments it may be possible to use the entire set of past key frames as a kind of map.

In addition to the extensions to key frame ICP mentioned above, the algorithm in general could be improved by further tuning it with respect to the vehicle dynamics. The navigation algorithm performance and stability depend on the proper adjustment of the ratio between the process model noise covariance and the measurement noise covariance. In practice, these values must be tuned together for best performance. In the simulation, initial performance of the ICP algorithm was unacceptable in anything but a hover due to large oscillations in

the navigation solution, which induced large oscillations in the control system output. Thus, the vehicle tended to oscillate around its commanded position, eventually becoming unstable when the scan matching algorithm eventually failed to find a match. After tuning the system the algorithm performed acceptably, as demonstrated in the simulation tests, although some oscillation is still visible in the data. With further tuning, and implementation of some of the extensions suggested above, the key frame ICP algorithm may perform as well as CoreSLAM, but with less computational effort.

The experimental results indicate that these two SLAM algorithms can potentially replace GPS for position updates, enabling rotorcraft UAVs to effectively navigate in indoor environments, without the requirement for a radio link to ground station computers. However, before implementation on an actual flying vehicle, the navigation system must be modified to compensate for time delay resulting from processing the laser scans. The method is somewhat straightforward, and requires keeping a brief history of the state information such that the measurement updates can be applied to states that are synchronized with the collection time of the laser scan measurements.

The implementation and testing of two efficient SLAM-based algorithms demonstrated that both have the potential for successful indoor navigation using the laser scanner, sonar, and IMU sensors. During the implementation of the algorithms, several important contributions were also made in the analysis of laser scan data with regard to information content. A novel recursive line segment extraction algorithm was developed that is an order of magnitude faster than competitive algorithms. In addition, a method for determining the quality of a specific local topology with regard to navigation was developed, providing a way for the navigation filter to "know" when it has little information with regard to its position due to a lack of features in the environment. This focus on fast, simple algorithms that still provide the information required for navigation is a significant improvement to the current approach to indoor aerial navigation. Rather than relying on popular algorithms in the robotics community that are accurate, but slow, this approach focuses on designing the navigation system from the ground up with the air vehicle in mind, thus enabling a completely self-contained onboard solution.

APPENDIX A

APPENDIX

A.1 Sonar Localization and Mapping

The sonar range sensors were tested to determine their characteristics and suitability for mapping indoor environments as discussed in Chapter 3, Section 3.1.3.2. A theoretical system for using sonar range readings to perform localization and mapping was developed and limited testing was done in simulation. Hardware difficulties and other schedule demands motivated a transition away from the system described in this appendix to the more capable system described in Chapter 4.

A.1.1 Sonar Mapping

The range measurements of the initial test discussed in Chapter 3 are repeated below in Figure 75. The results, similar for both sensors, illustrate a primary difference between a pulse-wave range measurement sensor (such as sonar or radar) and a point range measurement sensor (such as the IR sensors or a laser rangefinder). When facing a flat wall, the sonar measures the perpendicular distance to the wall, regardless of sensor orientation. A graph of the range readings versus angle show a series of straight lines, with some features such as corners and doorways observed. The horizontal lines correspond to the perpendicular distance from the sensor to major feature points in the room. The first feature detected during the sensor sweep was the right wall at 173cm. As the angle increased, the sensors began to lose range measurement to the right wall and pick up the front right corner at 264cm. The EZ4™ sensor picked up the corner first, while the EZ1™ sensor with the wider beam maintained a reading to the right wall longer before picking up the corner. As the sensor angle was increased, both sensors eventually picked up the front wall, which was at a distance of 198cm. The reading remained fairly constant until the sensors were turned far enough to pick up an open doorway in the front left corner. As the sensors were rotated

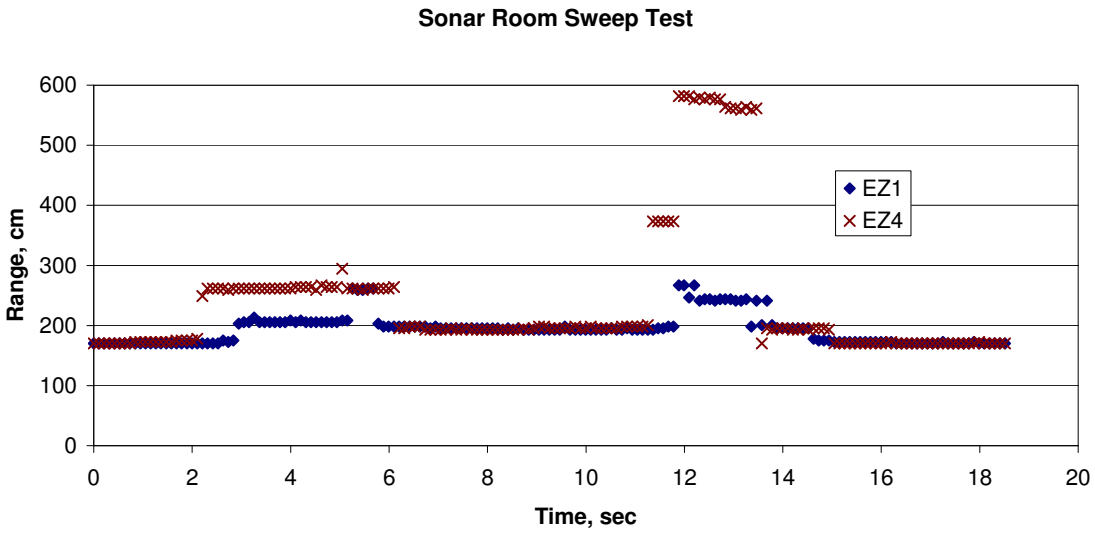


Figure 75: Distance measurements recorded using the EZ1™ and EZ4™ in simultaneous operation during a 180° sweep of a small room. Note the straight lines observed that correlate to the distance to the right, front, and left walls at 173cm, 198cm, and 173cm respectively. Some features such as corners and an open doorway are also observed.

further, they both eventually picked up the left wall at 173cm.

Although the sonar returns do not map to the wall locations directly, some features can be identified from the 2-D mapping. The arc that is visible from major geographic features represents the sonar wave pulse as it expands from the sensor. The sensors report the distance to the first return, which corresponds to the perpendicular distance to the walls and corners. Thus, the center of each visible arc in the 2-D plot represents the location of the sensor within the room. Given room dimensions and accurate heading information, it should be possible to localize a sonar sensor platform in a given room.

In order to determine room dimensions from observed range data, it is useful to consider the data shown in Figure 75. The straight lines representing distance to major geographical features are easily detected using a histogram approach. Figure 77 shows a histogram of the range data recorded during the test. The data bins are 10cm wide, so data in these bins represent an average value halfway between the bin label and the next lower bin. Looking at the histogram, the prominent features appear at the 180cm and 200cm bins. These represent measurements of approximately 175cm and 195cm, which correspond to

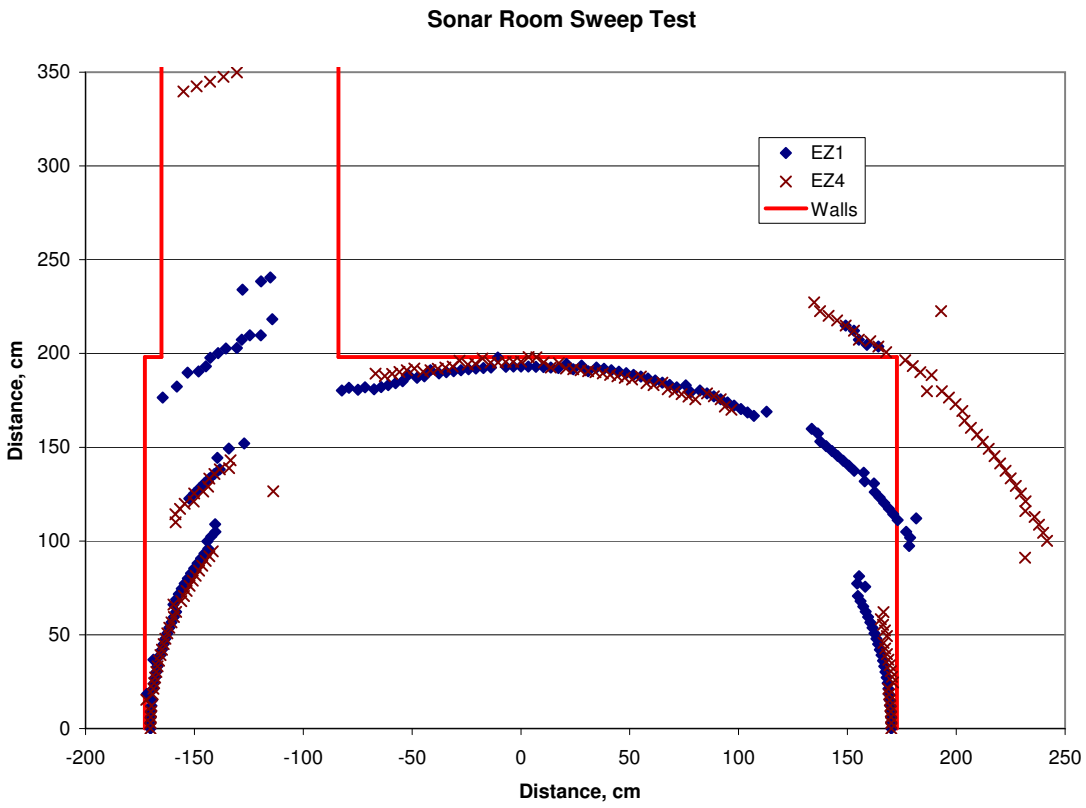


Figure 76: Distance measurements plotted using estimated sensor angle and measured range. Actual wall locations are show for reference. Note the EZ4™ more easily detects corners and open doorways than the EZ1™ .

the distance to the walls (173cm and 198cm respectively). An even more accurate estimate can be done by averaging the actual range measurements for the data that falls into the highest bins. As long as the data bin size is chosen wisely, however, the proposed method of using the middle of the bin for the measurement estimate is accurate enough for room size estimation.

Four sonar range sensors are arranged looking to the front, back, left and right of the aircraft as depicted in Figure 78. These sensors are read using the pulse-width method mentioned above via an interrupt pin on the ATMega128. They are used to measure the room dimensions and provide relative position information.

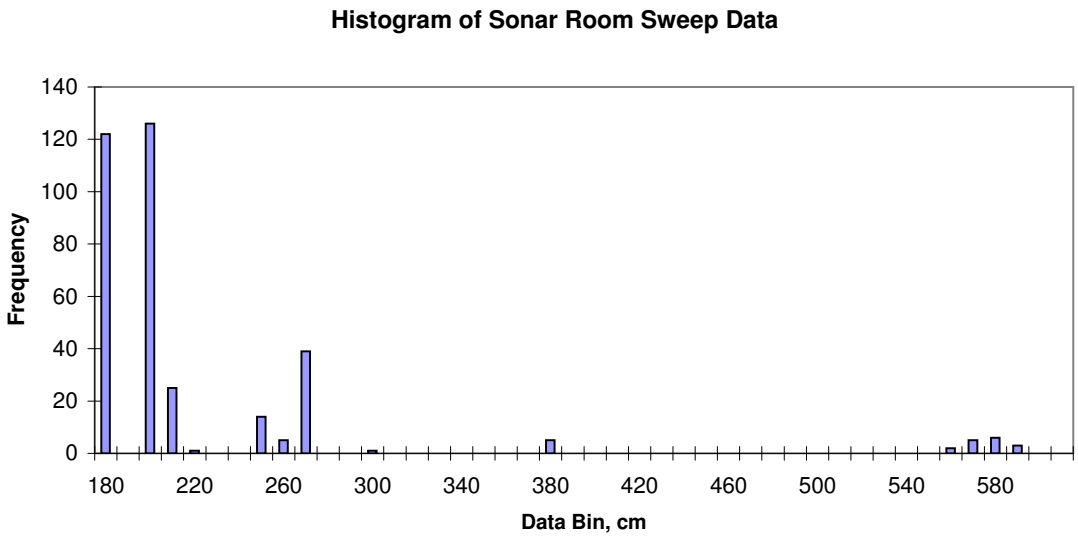


Figure 77: This histogram shows the frequency of measurements in each 10cm bin. The highest peaks represent room measurements of 175cm (average of 170-180cm bin) and 195cm (average of 190-200cm bin). Actual room measurements are 173cm and 198cm, respectively.

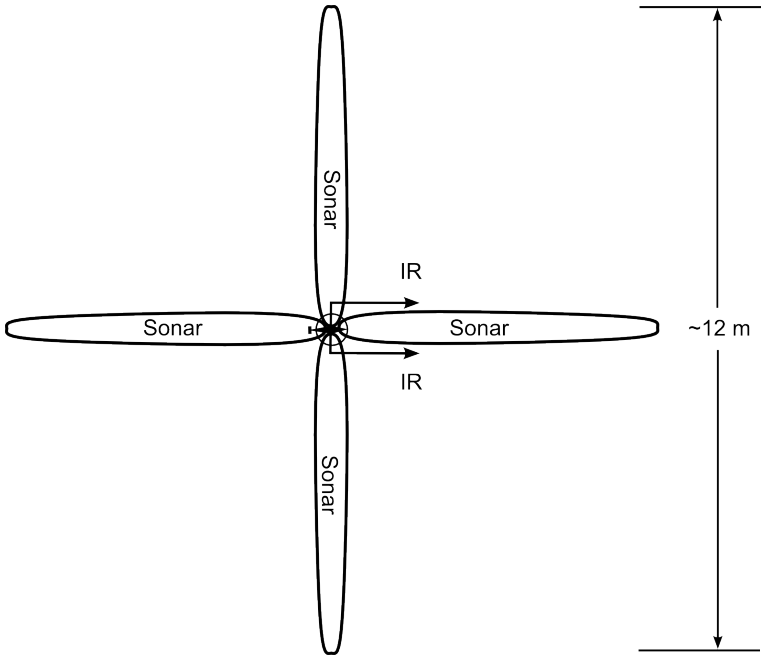


Figure 78: Range sensor layout for sonar mapping.

A.1.2 Navigation Algorithm

Once a reasonable range estimate is calculated, mapping and localization becomes possible. To use the range data collected by these low-cost IR and sonar sensors, additional assumptions are made to simplify the problem. By assuming that rooms are rectangular in shape and aligned to some Cartesian reference frame, the mapping problem is significantly reduced. The traditional method of measuring, recording, remeasuring, and adjusting map boundaries produces tens of thousands of data points, requiring extensive processing power and memory to be carried onboard the aircraft. By taking advantage of the regular structure exhibited by the majority of indoor environments, rooms can be mapped and stored using a 2-D parametric representation by determining characteristics such as dimension, location of room center, location of navigable openings, and location and radius of obstacles detected. Once initial mapping of a room has been completed, the map is stored and assumed correct while the vehicle remains in the room. The range sensors are then used to provide vehicle position relative to the stored map of the room being explored. Once the initial room has been explored to satisfaction, the vehicle flies through a detected opening into the next room and the process begins again. In this way, multiple rooms are explored with stored coordinates linking their mutual openings, and relative localization within each room is plotted on a global coordinate system based on the current room coordinates. The overall navigation scheme, described in further detail in sections that follow, is summarized in Figure 79.

A.1.3 Mapping

The first step in the process of mapping a room begins with the vehicle finding its way to the middle of the room. While this step is not strictly necessary, it is likely to provide the best view of the entire room. In addition, range measurements taken at regular angular increments are more spread out when the measurements are taken from the middle of a room. To find the middle of the room, the vehicle flies to a position where the “front” and “back” sensors read approximately the same range, while the “left” and “right” sensors also measure approximately the same range. The vehicle orientation is not important at

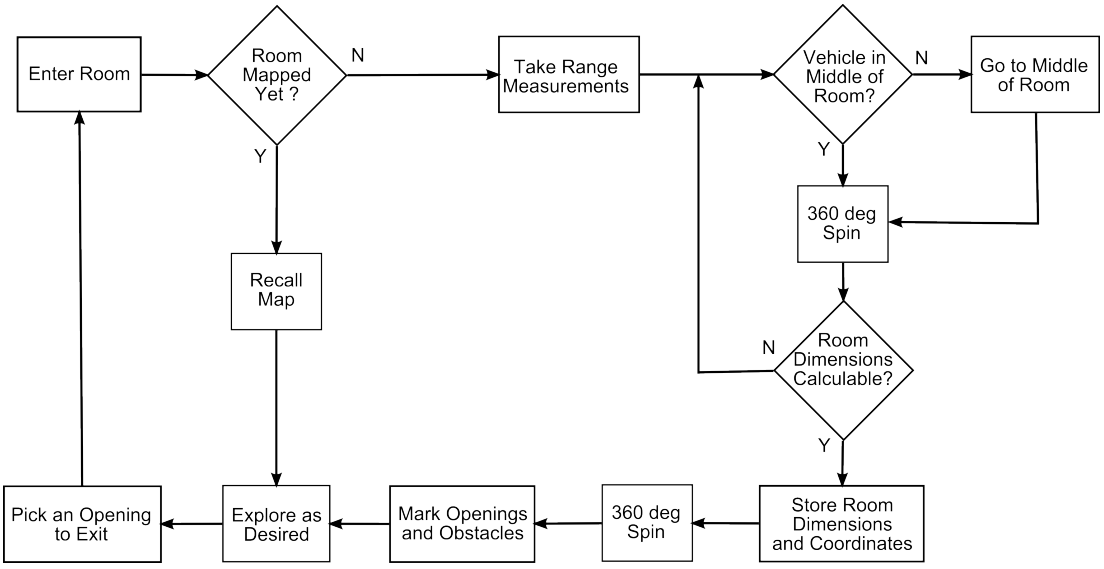
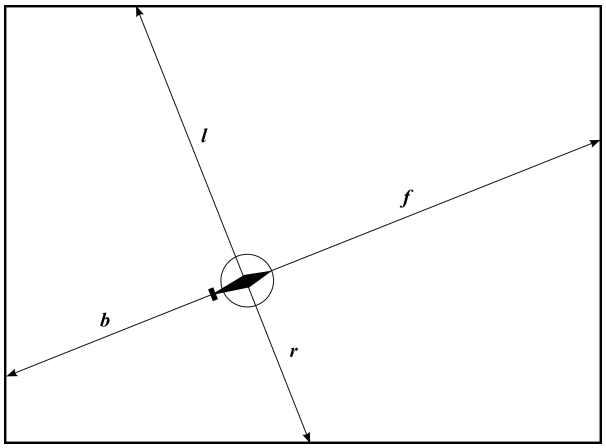


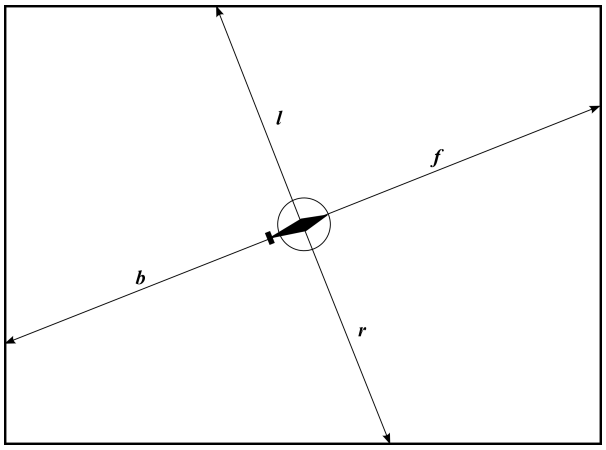
Figure 79: Navigation Scheme Flowchart. If the range information collected from the sensors is inconsistent with the the estimated vehicle position and orientation, the navigation sequence may be repeated as desired.

this time, so if the above range criteria cannot be suitably met, the vehicle undergoes a small yaw maneuver and tries again. Figure 80 shows how the vehicle uses its range sensors together to find the middle of a room.

Once the vehicle is satisfied that it is near the middle of the room, it performs a 360° yaw maneuver. During the maneuver, the rate gyro data is integrated to provide an angle measurement (relative to the original unknown orientation) for each range datum. The vehicle adds the range measured from the "front" and "back" sensors together to get a total longitudinal measurement, and it adds the range from the "left" and "right" sensors to get a total lateral range measurement. As a result, a full room sweep produces data similar to that shown in Figure 81. After the initial yaw sweep is accomplished, the data is analyzed and room dimensions are determined. For a rectangular room, a regular pattern of local minimum lateral and longitudinal range is clearly visible, repeated every 90°. Obstacles within the room produce short range readings as well, but these can typically be identified by comparing the two paired sensor ranges. In other words, if the two lateral or longitudinal sensors do not read approximately the same range, and obstacle is likely present at the shorter range and the reading at that angle should not be used for room dimensioning. If



(a) Initial position



(b) Final position

Figure 80: Moving to the center of the room by attempting to set $l{=}r$ and $f{=}b$.

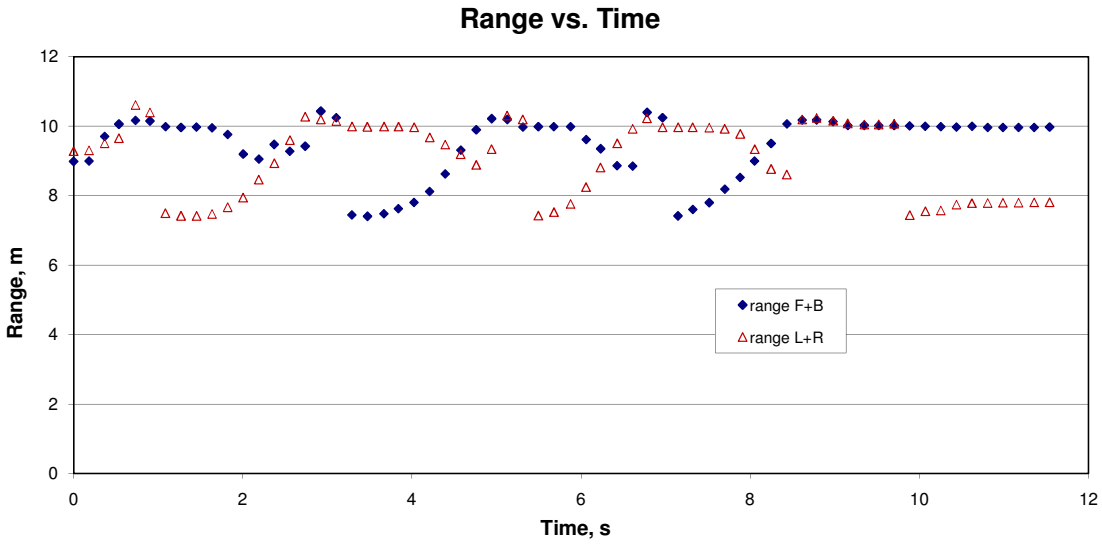


Figure 81: Simulated range measurement data during a room mapping 360° yaw sweep.

the vehicle drifts during the yaw maneuver, the error can be detected and corrected by comparing the range measurements identified at 90° intervals. The histogram for this data set is shown in Figure 82.

After the first room has been measured, the origin of a Cartesian reference frame is set at the center of the room. All future measurements can then be referenced to this coordinate system. A data structure is established to store the parameterized room data, which saves significant memory over storing all of the range data recorded (see Table 4). Parameters for the openings and obstacles are stored in substructures with their identifiers incremented based on the number of openings and obstacles detected (see Table 5). In this way, upon re-entering a room, the map can be fully recalled from the simple stored parameters. For example, `Room4.Opening2.Coords` would store the location of the second opening in fourth room explored. Likewise, `Opening2.ConnectsTo` would store the ID of the room adjoined to Room 4 via Opening 2. Obstacle location and size can be recalled in a similar manner.

Once the dimensions for a given room are calculated, another yaw sweep is performed, this time relative to the established coordinate system. The room dimensions are confirmed, and obstacles and potential doorways are located by looking for range readings that are closer or further than expected based on the room dimensions. A sweep of an empty

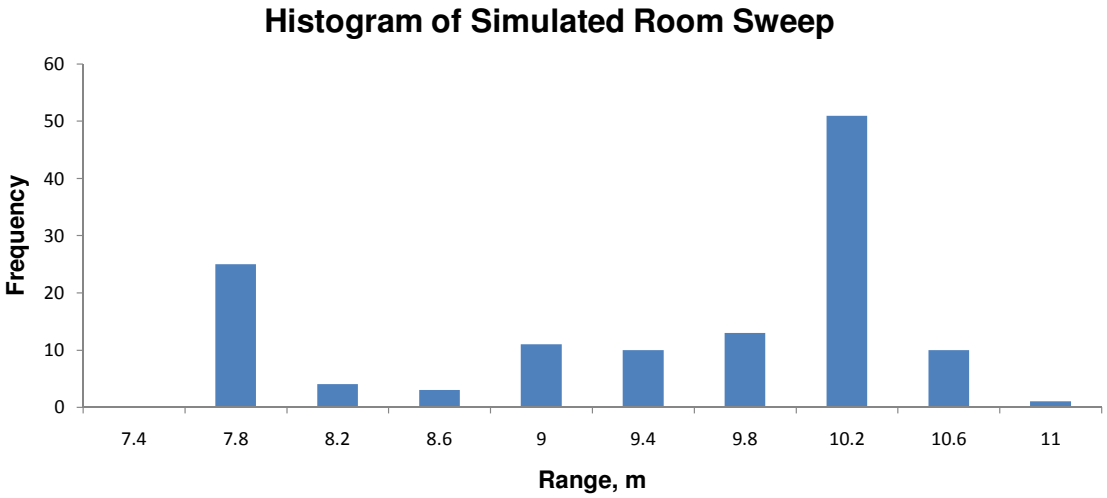


Figure 82: In this histogram of the simulated room measurement, the room dimensions are determined by observing the two data bins with the highest frequencies. In this test, bins 7.4m-7.8m and 9.8m-10.2m have the highest frequencies, corresponding to average measurements of 7.6m and 10.0m. Actual room dimensions were 7.3m and 9.8m.

Table 4: Data structure for storing room parameters

Label	Data Type
Room ID	string
Room Coordinates (x,y)	(integer,integer)
Room Dimensions (x,y)	(integer,integer)
Number of Obstacles	integer
Number of Openings	integer

room with two exits was performed in simulation. The data presented in Figure 81 can be plotted using the heading estimate from simulated gyro data. Once the room dimensions are calculated, the room data can be corrected for drift that occurs while the vehicle is performing the yaw sweep maneuver. Figures 83 and 84 show the room mapping plot before and after correcting for vehicle drift. Some rooms encountered may not be strictly rectangular. Rooms shaped similar to the one shown in Figure 85 are identified as two separate rooms, with an opening adjoining them.

Table 5: Data substructures for storing opening and obstacle data

(a) Substructure for openings.

Label	Data Type
Opening ID	string
Opening Coordinates (x,y)	(integer,integer)
Opening Width	integer
Room Connected To	string (Room ID)

(b) Substructure for obstacles.

Label	Data Type
Obstacle ID	string
Obstacle Coordinates (x,y)	(integer,integer)
Obstacle Size (x,y)	(integer,integer)

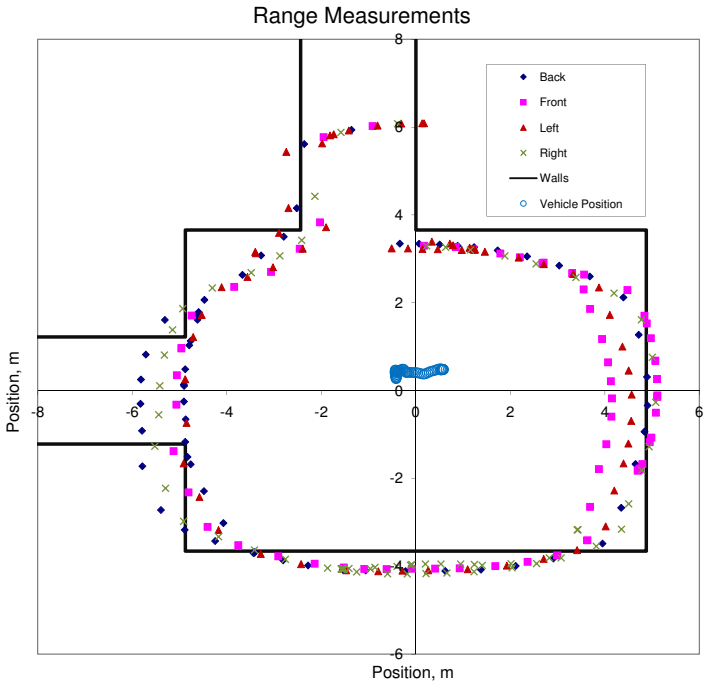


Figure 83: Simulated room sweep. The vehicle is rotated through a heading change of 360° while measuring range to the front, back, left, and right.

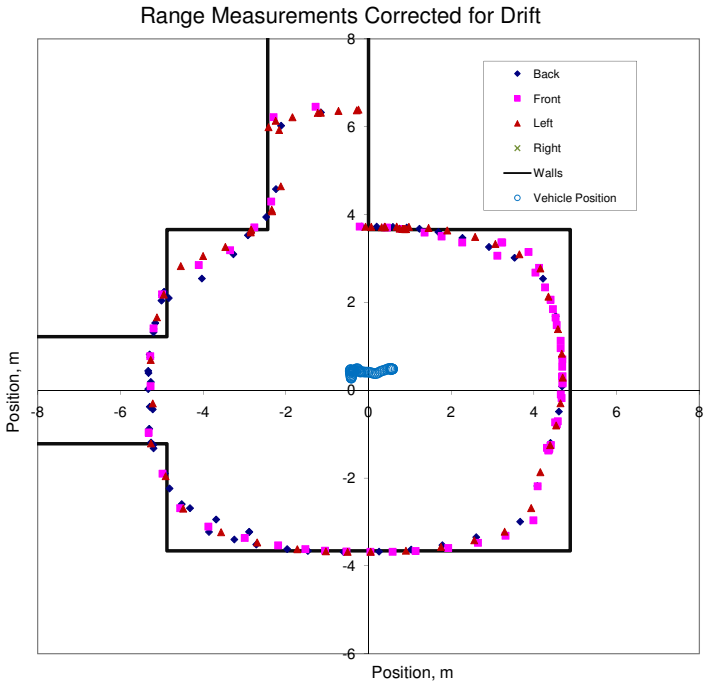


Figure 84: Simulated room sweep. The vehicle is rotated through a heading change of 360° while measuring range to the front, back, left, and right.

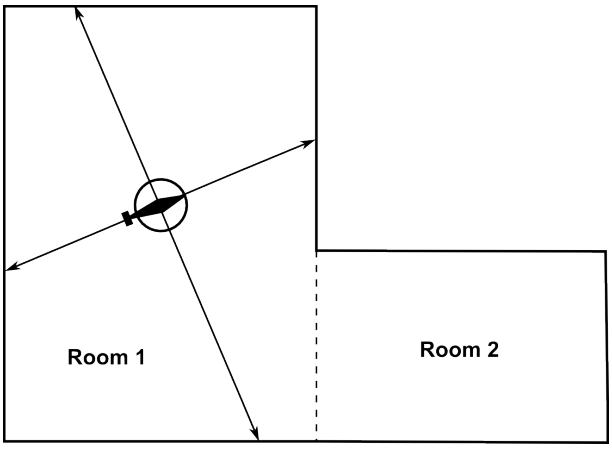


Figure 85: A room that is not rectangular would be broken up logically into two rooms with an adjoining opening between them.

A.1.4 Localization

Using the room data developed during the mapping stage, the vehicle can use its range sensors to determine its location within a given room. Theoretically, a given set of four range measurements does not determine a unique position within a rectangular room. However, if the vehicle maintains even a crude estimate of its position over time and the vehicle heading is correctly estimated to within $\pm 45°$, the possible positions for a given set of range measurements will be narrowed from four down to one. Location of obstacles and openings in the room give further clues to actual vehicle position. In addition to the sonar range finders, the two forward-facing IR sensors can be used to align the vehicle perpendicular to a wall, which allows "scanning" of a room while maintaining a known orientation by traversing from one end of the room to the other and measuring range to the opposite wall. In addition, using the two IR range sensors to maintain heading enables a less computationally expensive exploration routine whereby the vehicle follows a wall from room to room without actually knowing where it is. Employing this technique allows the vehicle to "escape" from a room if it cannot accurately determine its position or the position of openings in the room.

A.2 Sonar Mapping Simulation

A simulation was initially developed using the open-source software Blender [4]. In the simulation, coaxial helicopter vehicle dynamics are simulated, as well as the sonar and IR range sensors and a gyro to measure angular rate. The simulation allowed for easy configuration of different test environments, with variable room size and obstacle placement. In addition, the guidance and navigation algorithms were easily tested and refined. Mapping and localization algorithms were also tested using the simulation. Figure 28 in Chapter 3 shows a screen capture from the Blender simulation, and the data presented in Figures 83 and 84 were generated using this simulation.

A.3 Flight Test

Flight testing for the sonar mapping system was conducted using four EZ4™ sonar for longitudinal and lateral ranging, one EZ1™ sonar for altitude ranging, and two IR sensors for

heading estimation (see Figure 78). The sensor data was transmitted to a ground computer for processing, and servo commands were sent via a 2.4GHz hobby radio control transmitter. This setup facilitated testing of the sensors in flight under manual control before the onboard sensor processing and control algorithms were completed. During this phase of flight testing, characterization of the sonar and IR sensors was refined and altitude control and longitudinal obstacle avoidance were demonstrated. It was determined that in order to avoid interference between the sonar, each should be operated independently. With five sonar operating at 50ms per measurement, it takes 250ms to sample all of the sensors. However, a 4Hz update rate is not ideal for operating the altitude control loop, which is more sensitive to data rate. It is recommended to interleave the altitude measurement with the mapping sensors in the following manner: A-S1-A-S2-A-S3-A-S4, where "A" represents an altitude measurement, and "S1-S4" represent the four mapping sonar. Using this sampling order provides an altitude measurement at a rate of 10Hz, while mapping occurs at a slower rate of 2Hz. For longitudinal control in close proximity to a wall or obstacle, the IR sensors can be used since they operate at a rate of 25Hz. Unfortunately, due to problems with the sonar (possibly due to wiring or vibration), two of the sonar did not operate correctly using the first avionics package built and flown. As a result, research efforts were shifted to focus on building and testing another set of avionics specifically designed to test the wall-following guidance algorithm discussed in Chapter 3 in preparation for the upcoming IARC competition.

REFERENCES

[1] Achtelik, M., Bachrach, A., He, R., Prentice, S., and Roy, N., "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments," *SPIE*, vol. 7332, p. 10, Aug. 2009.

[2] Arras, K. O. and Siegwart, R. Y., "Feature extraction and scene interpretation for map-based navigation and map building," vol. 3210, pp. 42–53, SPIE, 1998. Available via: `http://link.aip.org/link/?PSI/3210/42/1`, DOI:10.1117/12.299565.

[3] Bailey, T. and Durrant-Whyte, H., "Simultaneous localization and mapping (SLAM): part II," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 3, pp. 108–117, 2006.

[4] Blender Foundation, "Blender foundation website," Jun 2010. Available via: `http://www.blender.org/`.

[5] Borges, G. A. and Aldon, M.-J., "Line extraction in 2d range images for mobile robotics," *Journal of Intelligent and Robotic Systems*, vol. 40, no. 3, 2004. DOI:10.1023/B:JINT.0000038945.55712.65.

[6] Borrmann, D., Elseberg, J., Lingemann, K., Nchter, A., and Hertzberg, J., "Globally consistent 3D mapping with scan matching," *Robotics and Autonomous Systems*, vol. 56, pp. 130–142, Feb. 2008. DOI:10.1016/j.robot.2007.07.002.

[7] Bosse, M., "An atlas framework for scalable mapping.". Available via: `http://www.robots.ox.ac.uk/~pnewman/papers/AtlasICRA2003.pdf`.

[8] Brooks, A. and Bailey, T., "HybridSLAM: combining FastSLAM and EKF-SLAM for reliable mapping." http://www.cas.edu.au/content.php/289.html?publicationid=550&displaypage=2. Available via: `http://www.cas.edu.au/content.php/289.html?publicationid=550&displaypage=2`.

[9] Bryson, M. and Sukkarieh, S., "Building a robust implementation of Bearing-Only inertial SLAM for a UAV," *Journal of Field Robotics*, vol. 24, pp. 113–143, 2007. Available via: `http://www.cas.edu.au/content.php/237.html?publicationid=361`.

[10] Caballero, F., Merino, L., Ferruz, J., and Ollero, A., "Vision-based odometry and slam for medium and high altitude flying uavs," *J. Intell. Robotics Syst.*, vol. 54, no. 1-3, pp. 137–161, 2009. DOI:http://dx.doi.org/10.1007/s10846-008-9257-y.

[11] Chen, Z., Samarabandu, J., and Rodrigo, R., "Recent advances in simultaneous localization and map-building using computer vision.," *Advanced Robotics*, vol. 21, pp. 233–265, Mar. 2007. DOI:10.1163/156855307780132081.

[12] Chowdhary, G. and Lorenz, S., "Control of a VTOL UAV via online paramter identification," in *AIAA Guidance Navigation and Control Conference*, (San Francisco,CA), 2005.

[13] Chowdhary, G., Ottander, J., Salaün, E., and Johnson, E. N., "Low cost guidance, navigation, and control solutions for miniature air vehicle in GPS denied environments," in *First Symposium on Indoor Flight*, 2009 International Aerial Robotics Competition, Jul 2009.

[14] Christophersen, H., Pickell, R. W., Neidhoefer, J. C., Koller, A. A., Kannan, S. K., and Johnson, E. N., "A compact guidance, navigation, and control system for unmanned aerial vehicles," *Journal of Aerospace Computing, Information, and Communication*, vol. 3, pp. 187–213, May 2006.

[15] Diosi, A. and Kleeman, L., "Fast Laser Scan Matching using Polar Coordinates," *The International Journal of Robotics Research*, vol. 26, no. 10, pp. 1125–1153, 2007. Available via: `http://ijr.sagepub.com/cgi/content/abstract/26/10/1125`, DOI:10.1177/0278364907082042.

[16] Dissanayake, M. W. M. G., Newman, P. and, D.-W. H., Clark, S., and Csorba, M., "A solution to the simultaneous localization and map building problem," *IEEE Transactions on Robotics and Automation*, vol. 17, 3, pp. 229–241, 2001.

[17] Durrant-Whyte, H. and Bailey, T., "Simultaneous localisation and mapping (SLAM): part i the essential algorithms," *Robotics and Automation Magazine*, vol. 13, p. 99110, 2006. Available via: `http://www.acfr.usyd.edu.au/homepages/academic/tbailey/papers/slamtute1.pdf`, DOI:10.1109/MRA.2006.1638022.

[18] Durrant-Whyte, H., "Equations for the prediction stage of the information filter." 2000. Available via: `http://www.acfr.usyd.edu.au/pdfs/training/Prediction%20Equations.pdf`.

[19] E-Sky, Inc., "E020 big lama website," Jun 2010. Available via: `http://www.twf-sz.com/english/products.asp?prodid=0291`.

[20] Eliazar, A. and Parr, R., "DP-SLAM: fast, robust simulataneous localization and mapping without predetermined landmarks," in *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, Morgan Kaufmann, 2003.

[21] Gelb, A., Joseph F. Kasper, J., Raymond A. Nash, J., Price, C. F., and Arthur A. Sutherland, J., *Applied Optimal Estimation*. The M.I.T. Press, 1974.

[22] Green, W. and Oh, P., "A MAV that flies like an airplane and hovers like a helicopter," in *Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 699–704, 2005.

[23] Grisetti, G., "Improved techniques for grid mapping with Rao-Blackwellized particle filters," 2006. Available via: `http://www.informatik.uni-freiburg.de/~stachnis/pdf/grisetti06tro.pdf`.

[24] Guivant, J. E. and Nebot, E. M., "Solving computational and memory requirements of Feature-Based simultaneous localization and mapping algorithms.," *IEEE Transactions on Robotics & Automation*, vol. 19, no. 4, pp. 749–755, 2003. DOI:Article.

[25] Hhnel, D., Schulz, D., and Burgard, W., "Mobile robot mapping in populated environments.," *Advanced Robotics*, vol. 17, pp. 579–597, Nov. 2003. DOI:10.1163/156855303769156965.

[26] International Aerial Robotics Competition, "IARC website," Jun 2010. Available via: http://iarc.angel-strike.com/.

[27] Johnson, E. N. and Kannan, S. K., "Adaptive trajectory control for autonomous helicopters," *Journal of Guidance Control and Dynamics*, vol. 28, no. 3, pp. 524–538, 2005.

[28] Johnson, E. N., Turbe, M., Wu, A., Kannan, S. K., and Neidhoefer, J. C., "Flight test results of autonomous fixed-wing transition to and from stationary hover," *Journal of Guidance Control and Dynamics*, vol. 31, pp. 358–370, March 2008.

[29] Kaess, M. and Dellaert, F., "Visual SLAM with a Multi-Camera rig," *College of Computing Technical Report*, p. 10, Feb. 2006. Available via: ftp://ftp.cc.gatech.edu/pub/gvu/tr/2006/06-06.pdf.

[30] Kannan, S. K., *Adaptive Control of Systems in Cascade with Saturation*. PhD thesis, Georgia Institute of Technology, Atlanta Ga, 2005.

[31] Kannan, S. K., Koller, A. A., and Johnson, E. N., "Simulation and development environment for multiple heterogeneous UAVs," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, (Providence, Rhode Island), August 2004.

[32] Kim, J. and Sukkarieh, S., "Autonomous airborne navigation in unknown terrain environments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 3, p. 10311045, 2004.

[33] Kim, J. and Sukkarieh, S., "Real-time implementation of airborne inertial-SLAM," *Robotics and Autonomous Systems*, vol. 55, pp. 62–71, 2007.

[34] Konolige, K., Bowman, J., Chen, J. D., Mihelich, P., Calonder, M., Lepetit, V., and Fua, P., "View-based maps," in *Proceedings of Robotics: Science and Systems*, (Seattle, USA), June 2009.

[35] Lalonde, J., Vandapel, N., and Hebert, M., "Data structures for efficient dynamic processing in 3-D.," *International Journal of Robotics Research*, vol. 26, no. 8, pp. 777–796, 2007. DOI:10.1177/0278364907079265.

[36] Leonard, J. J. and Durrant-Whyte, H. F., *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers, 1992. Available via: http://oe.mit.edu/~jleonard/pubs/ldw_kluwer1992.pdf.

[37] Leonard, J. and Durrant-Whyte, H., "Simultaneous map building and localization for an autonomous mobile robot," in *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, pp. 1442–1447 vol.3, 1991. DOI:10.1109/IROS.1991.174711.

[38] Lu, F. and Milios, E., "Robot pose estimation in unknown environments by matching 2d range scans," *Journal of Intelligent and Robotic Systems*, vol. 18, no. 3, pp. 249–275, 1997. DOI:10.1023/A:1007957421070.

[39] Lupton, T. and Sukkarieh, S., "Removing scale biases and ambiguity from 6DOF monocular SLAM using INS," in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation, vol. CD, pp. 3698-3703, 2008*, 2008. Available via: http://www.cas.edu.au/content.php/237.html?publicationid=547.

[40] Martínez, J., González, J., Morales, J., Mandow, A., and García-Cerezo, A., "Mobile robot motion estimation by 2d scan matching with genetic and iterative closest point algorithms," *Journal of Field Robotics*, vol. 23, no. 1, pp. 21–34, 2006. DOI:http://dx.doi.org/10.1002/rob.20104.

[41] MaxBotix® Inc., *LV-MaxSonar® -EZ4™ Data Sheet*, 2007.

[42] Newman, P. and Ho, K., "SLAM loop closing with visually salient features," 2005. Available via: `http://www.robots.ox.ac.uk/~pnewman/papers/NewmanHoICRA05.pdf`.

[43] nez, P. N., Vázquez-Martín, R., del Toro, J., Bandera, A., and Sandoval, F., "Natural landmark extraction for mobile robot navigation based on an adaptive curvature estimation," *Robotics and Autonomous Systems*, vol. 56, no. 3, pp. 247 – 264, 2008. Available via: `http://www.sciencedirect.com/science/article/B6V16-4P9SNFX-1/2/96b030b077e9734dc8f405ad40a8eccc`, DOI:DOI: 10.1016/j.robot.2007.07.005.

[44] Nguyen, V., Martinelli, A., Tomatis, N., and Siegwart, R., "A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 1929 – 1934, 2-6 2005. DOI:10.1109/IROS.2005.1545234.

[45] Nieto, J., Bailey, T., and Nebot, E. M., "Recursive Scan-Matching SLAM.". Available via: `http://www.cas.edu.au/content.php/237.html?publicationid=370`.

[46] Núñez, P., Vázquez-Martín, R., Bandera, A., and Sandoval, F., "Fast laser scan matching approach based on adaptive curvature estimation for mobile robots," *Robotica*, vol. 27, no. 03, pp. 469–479, 2009. Available via: `http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=5176428&fulltextType=RA&fileId=S0263574708004840`, DOI:10.1017/S0263574708004840.

[47] Okubo, Y., Ye, C., and Borenstein, J., "Characterization of the Hokuyo URG-04LX laser rangefinder for mobile robot obstacle negotiation," *SPIE*, vol. 7332, p. 10, 2009.

[48] Olson, E., *Robust and Efficient Robotic Mapping*. Doctoral thesis, Massachusetts Institute of Technology, June 2008.

[49] Pavlidis, T. and Horowitz, S., "Segmentation of plane curves," *Computers, IEEE Transactions on*, vol. C-23, pp. 860 – 870, aug. 1974.

[50] Pears, N. E., "Feature extraction and tracking for scanning range sensors," *Robotics and Autonomous Systems*, vol. 33, no. 1, pp. 43 – 58, 2000. Available via: `http://www.sciencedirect.com/science/article/B6V16-4118J6T-4/2/736447a948a1e27957a7753fb20c9d89`, DOI:DOI: 10.1016/S0921-8890(00)00089-0.

[51] Pradalier, C. and Sekhavat, S., "Simultaneous localization and mapping using the geometric projection filter and correspondence graph matching.," *Advanced Robotics*, vol. 17, pp. 675–690, Nov. 2003. DOI:10.1163/156855303769157018.

[52] Roberts, J. F., Stirling, T. S., Zufferey, J.-C., and Floreano, D., "Quadrotor using minimal sensing for autonomous indoor flight," in *3rd US-European Competition and Workshop on Micro Air Vehicle Systems (MAV07)& European Micro Air Vehicle Conference and Flight Competition (EMAV2007)*, 2007.

[53] Rusu, R. B., Marton, Z. C., Blodow, N., Dolha, M., and Beetz, M., "Towards 3D point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, pp. 927–941, Nov. 2008. DOI:10.1016/j.robot.2008.08.005.

[54] Saeedi, P., Lawrence, P. D., and Lowe, D. G., "Vision-Based 3-D trajectory tracking for unknown environments.," *IEEE Transactions on Robotics*, vol. 22, pp. 119–136, Feb. 2006. DOI:10.1109TRO.2005.858856.

[55] SHARP Corporation, *SHARP GP2Y0A02YK0F Distance Measuring Sensor Unit Data Sheet*, sheet no.: e4-a00101en ed., 2006.

[56] Shim, H., Koo, T. J., Hoffman, F., and Sastry, S., "A comprehensive study of control design for an autonomous helicopter," in *IEEE Conference on Decision and Control*, 1999.

[57] Simond, N. and Rives, P., "What can be done with an embedded stereo-rig in urban environments?," *Robotics and Autonomous Systems*, vol. 56, pp. 777–789, Sept. 2008. DOI:10.1016/j.robot.2007.11.008.

[58] Smith, R., Self, M., and Cheesman, P., "Esimating uncertain spatial relationships in robotics," in *Autonomous Robot Vehicles* (Cox, I. J. and Wilfong, G. T., eds.), pp. 167–198, New York, NY: Springer, 1990.

[59] Sobers, D. M., Chowdhary, G., and Johnson, E. N., "Indoor navigation for unmanned aerial vehicles," in *AIAA Guidance Navigation and Control Conference*, (Chicago, Illinois), p. 29, Georgia Institute of Technology, August 2009.

[60] Sparkfun™ Electronics, "Sparkfun™ electronics product website," Jun 2010. Available via: `http://www.sparkfun.com/`.

[61] Steux, B. and Hamzaoui, O. E., "CoreSLAM: a SLAM algorithm in less than 200 lines of C code," *Mines ParisTech*, p. 6, 2009. Available via: `http://infotrick.u7n.org/CoreSLAM/CoreSLAM.pdf`.

[62] Steux, B. and Hamzaoui, O. E., "Coreslam on openslam.org," Jun 2010. Available via: `http://www.openslam.org/coreslam.html`.

[63] Taylor, R. and Probert, P., "Range finding and feature extraction by segmentation of images for mobile robot navigation," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 1, pp. 95 –100 vol.1, 22-28 1996. DOI:10.1109/ROBOT.1996.503579.

[64] Thrun, S., Fox, D., Burgard, W., and Dellaert, F., "Robust monte carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2000.

[65] Vandorpe, J., Van Brussel, H., and Xu, H., "Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2d range finder," in *Robotics*

and Automation, 1996. Proceedings., 1996 IEEE International Conference on, vol. 1, pp. 901 –908 vol.1, 22-28 1996. DOI:10.1109/ROBOT.1996.503887.

[66] Vasconcelos, J., Cunha, R., Silvestre, C., and Oliveira, P., "A nonlinear position and attitude observer on se(3) using landmark measurements," *Systems & Control Letters*, vol. 59, no. 3-4, pp. 155 – 166, 2010. Available via: http://www.sciencedirect.com/science/article/B6V4X-4Y960RS-1/2/3c16f9592fd717273c22374f8130719c, DOI:DOI: 10.1016/j.sysconle.2009.11.008.

[67] Vázquez-Martín, R., Núñez, P., Bandera, A., and Sandoval, F., "Curvature-based environment description for robot navigation using laser range sensors.," *Sensors (14248220)*, vol. 9, no. 8, pp. 5894 – 5918, 2009. Available via: http://www.library.gatech.edu:2048/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=44086746&site=ehost-live.

[68] Wang, C., Thorpe, C., Thrun, S., Hebert, M., and Durrant-whyte, H., "Simultaneous localization, mapping and moving object tracking.," *International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, 2007. DOI:10.1177/0278364907081229.

[69] Wu, A. and Johnson, E., "Methods for localization and mapping using vision and inertial sensors," in *Proc. of AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2008.

[70] Wu, A., Johnson, E., and Proctor, A., "Vision-aided inertial navigation for flight control," in *Proc. of AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005.

[71] Yamaura, S., "Development of navigation algorithm for unmanned indoor flight vehicle by using 2d laser scanner." Georgia Tech AE 8900 Special Topics Report, 2010.

[72] Zhou, W., Miro, J., and Dissanayake, G., "Information-Efficient 3-D visual SLAM for unstructured domains," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1078–1087, 2008.